Towards a catalog standard for the relational model version 2

Veldwijk, R.J.

1991

document version
Publisher's PDF, also known as Version of record

Serie Research Memorandum

Towards a Catalog Standard for the Relational Model Version 2: A Manifesto

R.J. Veldwijk
R.B. Buitendijk
E.R.K. Spoor
M. Boogaard

Research Memorandum 1991-50
september 1991

vrije Universiteit *amsterdam*

Towards a Catalog Standard for the Relational Model Version 2

A Manifesto

Abstract
This manifesto presents a formal model of the structural and integrity aspects of the Relational Model of Data Version 2 (RM/V2) as presented by Codd. Because the modelling tool used is the Relational Model itself, this formal model is in fact a proposal for a relational catalog standard. It is argued that such a standard is sorely needed for many purposes, which include the obvious need for an RDBMS catalog standard, the assessment of RM/V2 and proposed enhancements to RM/V2, educational purposes, the assessment of RDBMSs and CASE-tools that claim to support the Relational Model, and the assessment of application database designs. The manifesto attempts to create a basis for further research in these directions.

1. INTRODUCTION

The Relational Model of Data is a model that enables application designers to capture reality in a formal way by means of relational data structures and constraints expressed in a Relational Language (RL). The Relational Model has been the dominant data model for some time and will probably remain so for long. Implementors of parts of the model in DBMS products have enjoyed great commercial success. Future implementation of neglected aspects of the model and additions to the model will probably consolidate this success. If so, application design with the Relational Model and application realization with RDBMS products will remain 'standard procedure' for the foreseeable future.

Although the Relational Model is a powerful tool for designing formal models of reality, the model itself is rarely expressed by means other than detailed and careful verbal description. Codd's latest book on version 2 of the model [CODD90], which is the foundation of this manifesto[1], is a good example of this practice. The result is a specification that lends itself to formalization, but this final step is transferred to the vendors of CASE and RDBMS products. The intuitive doubts one may have about their faithfulness to the model is strengthened by Codd's repeated complaints about RDBMS vendors.

An example of an attempt to formalize the Relational Model can be found in [DATE83]. Interestingly enough, Date uses a mathematical notation as his modelling tool. It would seem much more elegant if the Relational Model *itself* were used as such because (1) the Relational Model is supposed to be a powerful tool for specifying formal systems and because (2) the Relational Model is a formal system itself. Another relevant observation in this respect is that there is no essential difference between data and metadata [CURT81] [ROSS81] [VELD91b]. In other words, self-descriptiveness is an important aspect of 'good' data models.

An application of the self-modelling procedure with respect to a data model can be found in [NIJS89]. In this manifesto we aim to do the same for the Relational Model, more specifically Codd's latest version of this model

---

[1] References to [CODD90] are hereafter specified by their page or chapter number only.

(RM/V2). The resulting description can be used for a number of interrelated purposes. One can identify at least nine such purposes.

First, support of Data Base Management Systems for a data model requires formalization of that model in the DBMS software. Modelling the data model is thus a prerequisite for successful DBMS implementation. More specifically, RM/V2 demands that an RDBMS has a self-descriptive catalog, accessible to the user by means of an RL. The catalog is a special purpose relational database used by the RDBMS to store among others the structural and integrity aspects of application databases. The representation in the catalog of the catalog itself is equivalent to a relational representation of the Relational Model, at least of its non-manipulative aspects. The RM/V2 description to be presented can thus be used as input for RDBMS design.

Second, a RM/V2 catalog model should be used as a basis for a Relational Catalog standard. Lack of catalog standardization presently limits application portability and metadata accessibility, but may very well become a major bottleneck for achieving heterogeneously distributed DBMSs.

Third, a catalog model can be used as a means to measure the faithfulness to RM/V2 of both RDBMSs and Dictionary tools that claim to be useful in the relational database design process. A viable procedure to achieve this is to implement the catalog together with a database description that exploits every construct allowed by RM/V2[2]. If it is not possible to transfer the catalog contents to the RDBMS or Dictionary product, this product does not make it possible to exploit all features of RM/V2. If the contents of the RDBMS or Dictionary product cannot be transferred to the RM/V2 catalog, this product allows non-relational constructs, resulting in poor and non-portable database designs. In either case, the RDBMS or Dictionary product is not fully relational as far as the non-manipulative aspects of RM/V2 are concerned.

Fourth, the process of deriving a catalog model may reveal omissions, inconsistencies and ambiguities that have been overlooked by readers or writers of verbal descriptions of RM/V2.

Fifth, a catalog model is useful for assessing and discussing the merits of proposed enhancements of RM/V2, because the model combines formality with compactness and understandability. Furthermore, the effects of proposed enhancements on the complexity and the elegance of the catalog model itself provide a yardstick against which proposed enhancements can be measured. For example, it appears that the catalog model presented is quite complicated by the inability of RM/V2 to deal with generalization problems. The promised introduction of generalization support in RM/V3 [p.480] should thus have a profound and positive influence on the RM/V3 catalog model.

Sixth, a catalog model is a powerful tool for the education of database designers. A catalog model that is implemented in RDBMSs and CASE tools not only prevents the designer from devising 'forbidden' database models, but also offers insight in the Relational Model itself. Studying the model and querying the catalog self-description may be the easiest and most efficient way to gain insight in the Relational Model.

Seventh, besides preventing database designers to devise non-relational databases, a catalog model is a powerful tool for assessing (1) whether all RM/V2-concepts are covered by a database design and (2) whether this database design is internally consistent. The first criterion is satisfied if all catalog tables to be presented in this manifesto are filled, while

---

[2] As appendix A shows, the description of RM/V2 itself is not an example of such a database.

satisfaction of the second criterion requires all constraints in appendix B not to be violated. During the (probably long) period in which available RDBMS products will not fully support RM/V2, an implementation of the catalog model provides organizations with a means to manage the quality and portability of their application database designs. The catalog model thus becomes a temporary means to protect investments in non-fully relational DBMS environments.

Eighth, a catalog model permits the experienced database designer to design his application database as an extension to the catalog model. This blurring of data and metadata makes it possible to design extremely flexible applications that are cost-effective in environments characterized by unpredictable change, like management information systems [CURT81] [VELD90] [BOOG91]. For these implementations catalog standardization (see item 2) is exceptionally important.

Finally, a catalog model can serve as an aid in research directed at developing software tools and models that enable organizations to cope with evolutional change of their applications [SHNE82] [VELD91a]. The basis for this research is the observation that the extent to which the Relational Model supports logical data independence is quite poor.

Many of the arguments presented above are reminiscent of Codd's description of the use of data models in database management [CODD81]. In this respect it is unfortunate that Codd seems to be opposed to "*casting as much as possible of the system's behaviour into data structure*" [p.244]. This is not only in contrast with both database design theory (normalization theory) and practice, but it also obstructs crucial activities like catalog standardization [p.424] and catalog extension [p.278]. It would be very interesting to compare a proposed catalog standard according to Codd's design criteria to the catalog presented in this manifesto, which is based on a conventional database design approach.

In the following sections RM/V2 is described in an incremental fashion. This is necessary to keep the catalog model comprehensible and it provides the reader with an opportunity to skip aspects of the model he or she considers less interesting. The reader is supposed to have at least a basic understanding of the Relational Model and relational database design. Knowledge of Codd's book RM/V2 [CODD90] is helpful. Only the catalog model itself is presented and explained. The derivation of the model is not discussed. The constraints pertinent to the catalog model, i.e. entity integrity, referential integrity, domain integrity, A- and I-mark integrity, and the uniqueness of primary and alternate keys, are not listed as constraints. These constraints are implicitly defined by the contents of the self-descriptive catalog structure in appendix A.

The authors have tried to follow Codd's description all the way. However, in some cases, this results in a needlessly complicated and constrained catalog model. In these cases, the apparently better option has been preferred. Wherever this situation occurs, the deviation is amply discussed. Still, the authors do not pretend that the presented catalog is the best design possible. In some instances there are multiple modelling solutions, none of which is clearly 'the best'. This phenomenon points to weaknesses in RM/V2 (see also [MCGE76]), but discussion of these weaknesses is beyond the scope of this manifesto. Suffice to say here that a 'canonical' catalog standard of RM/V2 does not exist.

The discussion of the model is divided into a number of parts (see figure 1), to each of which a separate section is devoted.

Section	Subject
2	The Basic Model
3	Composite Columns and Composite Domains
4	Archives, Snapshots and Relational Assignments
5	Views
6	Indexes
7	Constraints and Triggered Actions
8	Built-in and User-Defined Functions
9	Synonym Names and Naming Rules
10	Database Users
11	Authorization
12	Audit Logging
13	Distributed Database
14	Database Statistics

Figure 1: Aspects of an RM/V2 catalog model

Section 15 provides an overview of the complete model. Section 16 contains some concluding remarks. Appendix A and B together provide a description of the catalog model in an unambiguous manner. Appendix A lists the definition of the catalog model in terms of its own database definition. It consists of a number of table contents of relations introduced in the body text. Appendix B reformulates the catalog's database constraints in SQL-statements wherever possible. Appendix C briefly lists some of the authors' criticisms with respect to RM/V2 itself.

2. THE BASIC MODEL

Basic to the Relational Model are concepts like 'relation', 'domain', 'attribute', 'key', 'entity integrity' and 'referential integrity'. Figure 2a depicts a relational schema that expresses these concepts. The boxes represent relations, the arrows represent foreign key to primary key references between these relations.

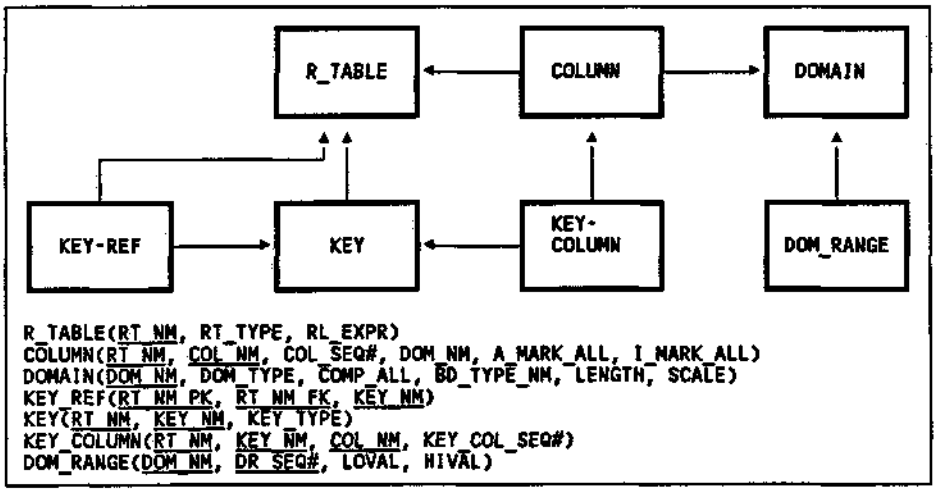


Figure 2a: A basic model of the Relational Model

An R_TABLE (or relation) in the Relational Model must have a unique name (RT_NM). An R_Table is either a base relation or a view (RT_TYPE). Views are defined in terms of other views and/or base relations by means of a statement in a relational language (RL_EXPR).

A COLUMN (or attribute) is identified by its name together with the name of the relation to which it belongs (RT_NM, COL_NM). Columns are assigned a sequence number for default presentation purposes. Every column belongs to a domain (DOM_NM). Columns that do not belong to the primary key can be allowed to be unknown but applicable (A_MARK_ALL), unknown but inapplicable (I_MARK_ALL) or both[3].

A DOMAIN must have a unique name (DOM_NM). If a primary key column draws its values from a domain, it is a primary domain (DOM_TYPE). An indicator (COMP_ALL) designates whether comparison of domain values is meaningful. Domains are extended data types based on DBMS dependent basic data types (BD_TYPE_NM) like 'character', 'number' and 'date'. Depending on the datatype the columns LENGTH en SCALE may be applicable.

Aside: Although Codd only speaks of "*a range of values*" permitted in columns belonging to a domain the authors believe multiple ranges of values are not forbidden. Admitting only one range of values per domain impairs the interpretation by catalog users of the meaning of the domain and leads to unnecessary domain constraints. Disallowing multiple ranges amounts to abolishing the view of domain as 'a pool of values' [CODD70].

The relation DOM_RANGE permits the specification of multiple ranges of values per domain by means of the columns LOVAL and HIVAL. If no range of values is specified for a domain, the range of values is only limited by the basic data type assigned to the domain.

A KEY is formed by one or more columns of one relation that identify a tuple in some relation. Primary and alternate keys identify tuples in the R-table to which these keys belong themselves. Foreign keys identify tuples in one or more R-tables designated by KEY_REF, if none of the foreign key columns contains an A-mark.

Aside: Codd makes no mention of the term candidate key or alternate key anymore. Alternate keys are considered to be user-defined constraints. However, he does specify that for each column the DBA should be able to specify that all values of a column are required to be distinct from one another [p.156], which amounts to alternate key support. Because full support for alternate keys leads to a less complicated catalog model such keys have been included.

KEY_COLUMN contains the columns constituting a key. A key column is identified by the combination of the identifiers of the R-tables KEY and COLUMN. The column KEY_COL_SEQ# provides the mechanism to couple a foreign key column to a primary key column. Composite primary/foreign key combinations must have matching sequence numbers for their columns. It is not possible to use the rule that matching primary and foreign keys should

---

[3] If the DBA specifies that a column is no longer to contain marks of any type, Codd requires the RDBMS to tolerate existing marks [p.253]. If the RDBMS is to be able to monitor its own integrity, changing a 'marks-allowed' indicator from 'YES' to 'NO' requires the RDBMS to mark the marked tuples as '*inserted before disallowance of marks*'.

have the same domain because more than one column in a key can belong to a given domain. A foreign key references at least one R-table. Every such reference is expressed by a KEY_REF-tuple.

It appears that the core aspects of the Relational Model can be expressed quite concisely and, at least in our opinion, elegantly by means of the Relational Model itself. Nevertheless, there remain a number of constraints that are not enforced by the data structure of figure 2a. These constraints have to be expressed in an ad hoc manner by means of RL and stored in the catalog (see section 7). Figure 2b lists the constraints pertinent to the basic model of figure 2a. In appendix B these constraints are formally expressed in SQL.

BM01- Every tuple in R_TABLE is referenced by at least one COLUMN-tuple.
BM02- No view definition (RL_EXPR) is directly or indirectly recursive.
BM03- Every tuple in R_TABLE for which RT_TYPE = 'BASE' or 'CHECKOUT' (see section 4) is referenced by a KEY-tuple having KEY_TYPE = 'PRIMARY'.
BM04- The subset of KEY-tuples for which KEY_TYPE = 'PRIMARY' does not contain duplicate values for RT_NM.
BM05- Every tuple in KEY having KEY_TYPE = 'FOREIGN' is referenced by at least one KEY_REF-tuple, while other tuples in KEY are not referenced by KEY_REF-tuples.
BM06- Every tuple in KEY_REF references a tuple in R_TABLE that is referenced by a tuple in KEY having KEY_TYPE = 'PRIMARY'.
BM07- Every tuple in KEY is referenced by at least one KEY_COLUMN-tuple.
BM08- KEY-tuples with KEY_TYPE = 'PRIMARY', that are referenced via KEY_REF by a KEY-tuple with KEY_TYPE = 'FOREIGN', are referenced by pairs of KEY_COLUMN-tuples with corresponding values for KEY_COL_SEQ#, referencing COLUMN-tuples with the same value for DOM_NM.
BM09- No two KEY-tuples, for which KEY_TYPE = 'PRIMARY', 'FOREIGN' or 'ALTERNATE', can be referenced by identical sets of KEY_COLUMN-tuples, except for key combinations in which the values of KEY_TYPE are either 'PRIMARY' and 'FOREIGN' or 'ALTERNATE' and 'FOREIGN'.
BM10- The column RL_EXPR in R_TABLE contains an I-mark if and only if RT_TYPE = 'BASE'.
BM11- The values of COL_SEQ# in the set of COLUMN-tuples with the same value for RT_NM are numbered consecutively up from 1 to the number of tuples in the set.
BM12- The values of KEY_COL_SEQ# in the set of KEY_COLUMN-tuples with the same value for RT_NM and KEY_NM are numbered consecutively up from 1 to the number of tuples in the set.
BM13- No tuple in COLUMN that is referenced by a tuple in KEY_COLUMN that references a tuple in KEY having KEY_TYPE = 'PRIMARY' has a 'YES' value for either A_MARK_ALL or I_MARK_ALL.
BM14- No tuple in COLUMN that is referenced by a tuple in KEY_COLUMN that references a tuple in KEY having KEY_TYPE = 'FOREIGN' has a 'YES' value for I_MARK_ALL.
BM15- Every value of LOVAL is less than or equal to the value of HIVAL in the same DOM_RANGE-tuple.

Figure 2b: User-defined constraints on the basic model (continued on the next page)

BM16- The values of DR_SEQ# in the set of DOM_RANGE-tuples with the same value for DOM_NM are numbered consecutively up from 1 to the number of tuples in the set.
BM17- Tuples in DOM_RANGE with the same value for DOM_NM do not have overlapping ranges.
BM18- Every value of LOVAL or HIVAL belongs to the basic data type of the DOMAIN-tuple referenced by the DOM_RANGE-tuple.

Figure 2b: User-defined constraints on the basic model (continued)

The majority of these constraints require no elaboration and are easily expressed in RL (see appendix B). Exceptions are constraints BM08 and BM09, which are quite complex, and constraints BM02 and BM18, which cannot be expressed in RL, but must be implemented in a host language. Constraints BM04, BM11 and BM12 will be trivially modified in section 10 (see Figure 10c).

3. COMPOSITE COLUMNS AND COMPOSITE DOMAINS

RM/V2 provides support for composite columns and composite domains. Composite columns and domains offer an easy way to express joins in which multiple columns are involved. Figure 3a depicts a schema that expresses these concepts. Single lined boxes represent relations presented previously. Double lined boxes represent new relations. Relevant columns that have been presented previously are represented in lower case.

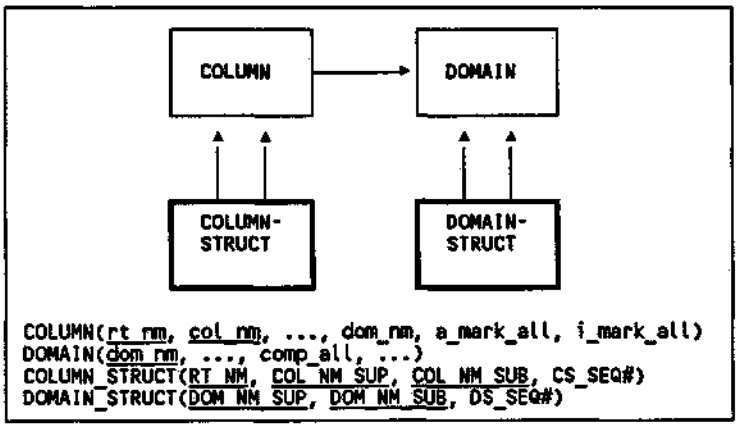


Figure 3a: Catalog support for composite columns and domains

The similar treatment of elementary and composite columns and domains is reflected in the catalog model. For every composite column or domain there exists a tuple in COLUMN or DOMAIN respectively. The structure of composites is reflected in the R-tables COLUMN_STRUCT and DOMAIN_STRUCT. Columns RT_NM and COL_NM_SUP designate the composite column, columns RT_NM and COL_NM_SUB designate an elementary column, and CS_SEQ# specifies the sequence number of the elementary column in the composite column's specification. The R-table DOMAIN_STRUCT provides the same function for composite domains.

Figure 3b (overleaf) lists the constraints with respect to composite columns and domains.

CM01- The set of tuples in COL_NM_SUP, projected over USER_NM, RT_NM and COL_NM_SUP, is disjunct from the set of tuples projected over USER_NM, RT_NM and COL_NM_SUB.
CM02- The set of tuples in DOM_NM_SUP, projected over DOM_NM_SUP, is disjunct from the set of tuples projected over DOM_NM_SUB.
CM03- The values of CS_SEQ# in the set of COLUMN_STRUCT-tuples with the same value for COL_NM_SUP are numbered consecutively up from 1 to the number of tuples in the set.
CM04- The values of DS_SEQ# in the set of DOMAIN_STRUCT-tuples with the same value for DOM_NM_SUP are numbered consecutively up from 1 to the number of tuples in the set.
CM05- For every composite COLUMN-tuple there exist corresponding pairs of COLUMN_STRUCT and DOMAIN_STRUCT-tuples for which:
- CS_SEQ# and DS_SEQ# are equal;
- DOM_NM_SUP in DOM_STRUCT equals DOM_NM in the COLUMN-tuple designated by COL_NM_SUP;
- DOM_NM_SUB in DOM_STRUCT equals DOM_NM in the COLUMN-tuple referenced by the COLUMN_STRUCT-tuple.

CM06- If columns constituting a composite column have different values for A_MARK_ALL or I_MARK_ALL, these columns are assigned I_MARKs for the composite column tuple, while the values are inherited by the composite column tuple if the elementary columns have similar values for A_MARK_ALL or I_MARK_ALL. Elementary columns cannot accept marks for A_MARK_ALL and I_MARK_ALL.
CM07- If domains constituting a composite domain have different values for COMP_ALL, this column is assigned the value 'NO' for the composite domain tuple, while the value is inherited by the composite domain if the elementary domains do not have different values for COMP_ALL.
CM08- DOMAIN-tuples must contain an A-mark for BD_TYPE_NM and I-marks for LENGTH and SCALE if and only if they are referenced by DOMAIN_STRUCT-tuples by means of the column DOM_NM_SUP.
CM09- A KEY_COLUMN-tuple must not reference a composite COLUMN-tuple.

Figure 3b: Constraints for composite columns and domains

Constraints CM01 and CM02 specify that composites of composites are not allowed to exist. These rules are specified because Codd does not see a practical need for such constructs [p.38]. It is interesting to note that support for composite hierarchies would result in different constraints (i.e., no loops in the hierarchies) that cannot be expressed in RL, and thus would be cumbersome to support. If support for 'recursive join' will be included in RM/V3, composites of composites will be supported easily.

Constraint CM09 is introduced to prevent the database user from specifying redundant indexes (see section 6).

Constraints CM01 and CM03 will be trivially modified in section 10 (see figure 10c). Constraints CM06 to CM08 are neither mentioned by Codd nor implied by his discussion of the subject, but seem reasonable nevertheless.

4. ARCHIVES, SNAPSHOTS, RELATIONAL ASSIGNMENTS AND CHECKOUTS

Apart from base relations and views, relations in RM/V2 can take the form of archives, snapshots, relational assignments, checkouts and replicas. An archive is a copy of a base R-table or a view, usually made for performance or safety purposes.

A snapshot is a relation valued expression in RL that is stored as a relation, and is accessible to authorized users. Updates on base relations after the creation of the snapshot are not reflected in the snapshot, like is the case with archives. Unlike an archive, a snapshot is accessible by terminal users and application programs. The differences between a snapshot and a relational assignment are:

1- a snapshot must have a primary key, while the designation of a primary key to a relational assignment is optional;
2- a relational assignment is not accessible to other database users and is discarded at the end of the session in which it is created.

A checkout is a special kind of snapshot, used in engineering environments [p.240]. A checkout enables the user to make a copy of part of the database for the purpose of making design changes to an object of interest, like a piece of machinery. The RDBMS marks the copied rows to ensure that these rows are not updated. At the time the checkout is returned, the RDBMS resets the marks.

Replicas are used in connection with distributed database management. Therefore, discussion of replicas is postponed to section 13. Figure 4a depicts the extensions to the catalog necessary to support the constructs discussed.

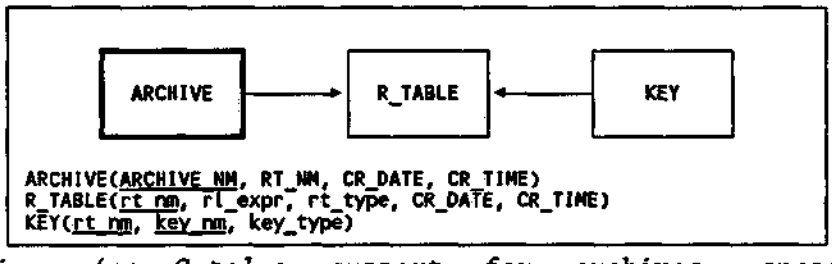


Figure 4a: Catalog support for archives, snapshots and relational assignments

The column RT_type distinguishes between snapshots, relational assignments and checkouts. Archives are represented by a separate R-table. Every archive has an associated source relation (RT_NM). An archive has a creation date (ARCHIVE.CR_DATE) and time (ARCHIVE.CR_TIME). Archiving and reactivating archives can be done by hand or by means of a clock-triggered action (see section 7). Every snapshot has a definition in RL (RL_EXPR) and a creation date (R_TABLE.CR_DATE) time (R_TABLE.CR_TIME). The columns CR_DATE and CR_TIME are relevant for any R-table, not just for snapshots. Figure 4b lists the constraints pertinent to archive and snapshot support.

AR01- The combination of CR_DATE and CR_TIME in ARCHIVE does not exceed the current system date and time.
AR02- The combination of CR_DATE and CR_TIME in R_TABLE does not exceed the current system date and time.
AR03- Tuples in R_TABLE for which RT_TYPE <> 'BASE', 'VIEW' or 'CHECKOUT' are not referenced by KEY-tuples for which KEY_TYPE = 'FOREIGN'.

Figure 4b: Constraints for archives, snapshots and relational assignments

Constraints AR03 and AR04 are introduced to prohibit the specification of foreign keys for snapshots and relational assignments. Although Codd does not mention this requirement, prohibition of such keys for snapshots follows from the fact that referential integrity cannot be enforced, because changes to the database are never reflected in the snapshots' contents.

Checkouts *must* have primary keys, while archives and snapshots *can* have them. With regard to archives the DBMS software may presume the primary key to be inherited from the source R-table. Note that the DBMS software must take action with regard to existing archives, whenever changes are made with respect to the source relation by means of DDL. Codd does not describe the action to be taken by the RDBMS in this case. Finally, we indicate that the relation between the R-tables ARCHIVE and R_TABLE will be changed in section 13, due to the introduction of database sites. This will result in the transfer of the columns CR_DATE and CR_TIME to another R-table and a trivial change in constraint AR02.

5. VIEWS

Although general support for views has already been provided in the basic model presented in section 2, supporting all requirements concerning views requires specific extensions to the catalog model. These extensions are depicted in Figure 5a.

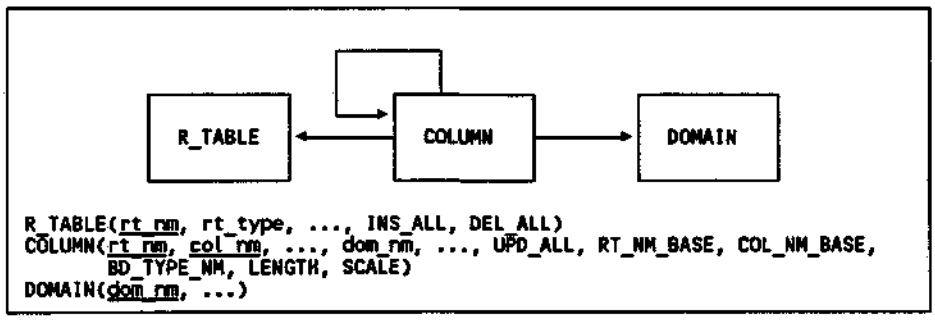


Figure 5a: Catalog support for views

The fact that views may have primary and foreign keys has already been covered by the basic model. Therefore, the catalog tables containing information about keys, key-columns and key-references are not shown.

The relation R_TABLE is extended with columns that designate whether inserts (INS_ALL) and deletes (DEL_ALL) on the view are allowed. For the same reason, the column UPD_ALL is added to COLUMNS. The values of these columns are determined at view definition time by a special algorithm. Depending on these values, the RDBMS software accepts or rejects attempts to modify the database via the view.

At view definition time, the DBMS software tries to determine a source column for every view column. If such a column exists, a reference to it is included in the catalog relation COLUMN by means of the columns RT_NM_BASE and COL_NM_BASE.

At view definition time, the domains to which the view columns belong are, if possible, determined by the RDBMS software. If the domain cannot be determined, only the basic data type of the view column is stored in the catalog. This means that some COLUMN-tuples cannot be required to reference DOMAIN-tuples anymore. Because each column must have a basic data type, the DOMAIN columns BD_TYPE_NM, LENGTH, and SCALE have now been duplicated in COLUMN.

Because views are often unnormalized relations, it is not guaranteed that an updatable view can always be updated freely. Codd states [p.302] that it is the responsibility of the DBA to declare for each base relation and view whether it is fully normalized. The information in this declaration is saved in the catalog. For this reason he distinguishes a number of generalized constraints, to be supported by the RDBMS. Discussion of this constraint support feature is deferred to section 7.

Aside: It is not clear why Codd is so concerned about possible update anomalies, because updates on views are translated to updates on base R_Tables. Update anomalies resulting in unpermitted database states cannot be effected because in such cases one or more constraints will be violated (see section 7). Specification by the DBA of constraints at view level seems redundant as well as error-prone.

A last aspect of view support concerns so called 'view interpretation functions' that can be used to solve update ambiguities in views that are unions of base R_Tables. Discussion of these functions is deferred to section 8.

The constraints that apply to the described extensions to the catalog are listed in Figure 5b.

VW01- An R_TABLE-tuple contains I-marked values for INS_ALL and DEL_ALL and a COLUMN-tuple contains I-marked values for UPD_ALL if and only if RT_TYPE <> 'VIEW'.
VW02- For every COLUMN-tuple, the columns RT_NM_BASE and COL_NM_BASE are both I-marked or not marked.
VW03- Every COLUMN-tuple for which the column RT_NM_BASE is not I-marked, references an R_TABLE-tuple for which RT_TYPE contains the value 'VIEW'.
VW04- COLUMN-tuples for which the column DOM_NM is A-marked reference R_TABLE-tuples for which RT_TYPE = 'VIEW', 'SNAPSHOT' or 'REL_ASGN'.
VW05- COLUMN-tuples, for which the column DOM_NM is not marked, have values for BD_TYPE_NM, LENGTH, and SCALE that are equal to the values of similarly named columns in the referenced DOMAIN-tuple.

Figure 5b: Constraints for view support

Constraint VW02 will be trivially modified in section 10 (see figure 10c).

6. INDEXES

Indexes are non-relational constructs that may only affect the performance of DML-operations on a relational database. Codd does not require a an RM/V2 DBMS to support conventional indexes. He only requires the catalog to support the construct if the DBMS provides the DBA with indexes (feature RE-14).

Surprisingly, Codd does require RDBMS vendors to support the non-relational construct of the *domain-based* index (feature RD-7). This type of index contains values of columns that belong to one specific domain. A subset of the columns belonging to the domain can be included in the index. Domain based indexes are ideally suited to speed up equi-joins.

Figure 6a expresses the extensions to the catalog to support both conventional and domain-based indexes.

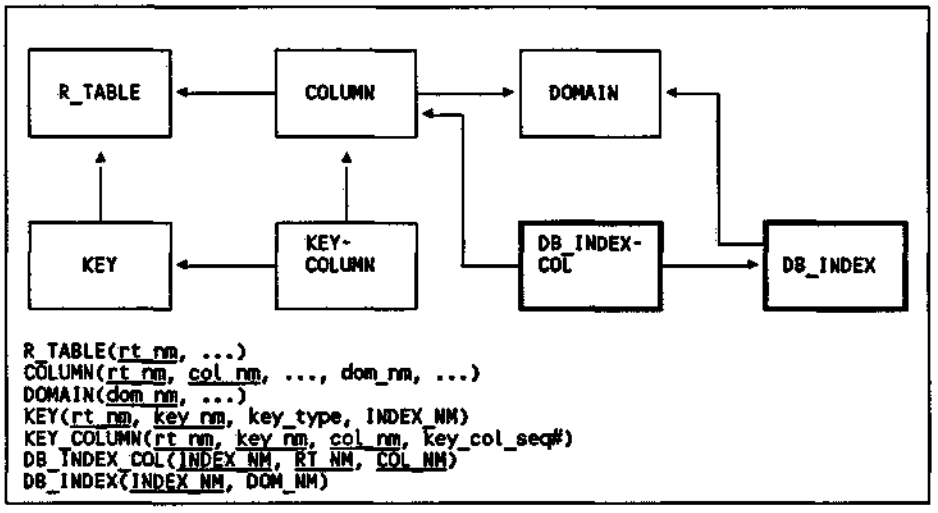


Figure 6a: Catalog support for indexes

The catalog treats domain based indexes and conventional indexes differently. It is assumed that it is possible to define more than one domain based index on a specific domain. To achieve this, the relations DB_INDEX and DB_INDEX_COL are added to the catalog. DB_INDEX contains the existing domain-based indexes and designates the domain to which the index belongs. DB_INDEX_COL contains the columns constituting a domain-based index. The catalog structure permits the allocation of one column to several indexes based on the column's domain.

Conventional indexes are not tied to domains but to R-tables. In most cases conventional indexes will coincide with primary, foreign or alternate keys, because selections, joins and orderings will normally be based on such keys. Therefore, it seems a good idea to extend the use of the catalog structures that support these keys. To achieve this, the column INDEX_NM is added to the relation KEY. INDEX_NM may be I-marked. Indexes on columns or combinations of columns that do not constitute a primary, foreign or alternate key are supported by permitting KEY-tuples for which KEY_TYPE = 'INDEX'. Of course, INDEX_NM itself constitutes an alternate key for the R-table KEY.

Figure 6b lists the user-defined constraints that apply to the extension of the catalog model.

IX01- The projections of DB_INDEX and KEY on their column INDEX_NM are disjunct.
IX02- Every DB_INDEX-tuple is referenced by at least two DB_INDEX_COL-tuples.
IX03- Every tuple in DB_INDEX_COL references the same DOMAIN-tuple via COLUMN and DB_INDEX.
IX04- The projection of the columns RT_NM and COL_NM in DB_INDEX_COL for one given value of INDEX_NM is not a subset of another such projection for another value of INDEX_NM.
IX05- DB_INDEX_COL tuples do not reference R_TABLE-tuples (via COLUMN) that have the value 'VIEW' for their column RT_TYPE.
IX06- KEY-tuples for which the column INDEX_NM is not I-marked reference R_TABLE-tuples for which RT_TYPE <> 'VIEW'.
IX07- KEY-tuples for which KEY_TYPE = 'INDEX' do not have a I-mark for their column INDEX_NM.
IX08- For the subset of KEY-tuples for which INDEX_NM is not I-marked, the projection of the columns RT_NM, COL_NM and KEY_COL_SEQ# in KEY_COLUMN for one given value of KEY_NM is not a subset of another such projection for another KEY-tuple.

Figure 6b: Constraints for indexes

Constraint IX01 forbids domain-based and conventional indexes to have the same index name, as indicated by Codd's feature RE-17. Constraints IX04 and IX08 forbid the specification of redundant indexes. Such indexes do not improve retrieval performance and impair update performance. Allowing the user to specify a redundant index is thus a violation of the idea that lies at the root of index support.

The purpose of domain-based indexes is to speed up joins. Joins are often performed over more than one column. To support such joins effectively, domain-based indexes must be applied to composite columns and domains.

Constraints IX01, IX04 and IX08 will be trivially modified in section 10 (see figure 10c).

7. CONSTRAINTS AND TRIGGERED ACTIONS

There are two problems involved in defining a catalog standard that supports constraints and triggered actions in RM/V2. The *first* problem is the occurrence of generalization problems in this area. Although constraints require a uniform treatment for many purposes, Codd distinguishes five different classes of constraints, each with its own peculiarities, that should be supported by a RM/V2-catalog. If, as promised, RM/V3 will offer generalization support, the part of the catalog to be described in this section will become much easier to comprehend and implement. The current lack of support for generalization constructs provides us with various design options, none of which is obviously 'correct' [VELD90] [BOOG91].

A *second* problem is caused by the fact that although constraints express much of the semantics of a database design (including the catalog design), it seems impossible to fit them in a finite number of classes. Codd rightly

claims that the Relational Model is superior over many other data models [CODD81] because it includes an RL by which most constraints can be expressed. Unfortunately, the semantic content of constraints expressed in this way cannot be found in the catalog by either the DBMS or a human user. This is in contrast to constraints that can either be expressed in R-table structures (like functional dependencies that are a consequence of the primary key) or explicitly in the RM/V2 catalog (like a primary key uniqueness constraint). Somewhere, one has to draw an arbitrary line between generalized and non-generalized constraints. Attempts of the DBMS to interpret the meaning of the latter type of constraints means blurring the distinction between programs and data and is at least a very complicated and probably an impossible task.

As noted in the introduction, Codd takes a rather negative position with regard to the use of data structures to capture reality. In his RM/V2 book he specifies that constraints not only *can* be expressed linguistically, but they also *should* be expressed linguistically [p.244]. At the same time he requires a fully relational DBMS catalog to support the various types of dependencies as generalized constraints.

The authors feel that it would have been better to specify that every constraint should be either explicitly expressed in RL or be expressible in RL by the RDBMS software on the basis of the information in the catalog. Nevertheless, the authors have followed Codd's specifications, except with respect to alternate keys, for the same reason as in section 2.

Figure 7a shows the extensions to the catalog necessary to support constraints and triggered actions.

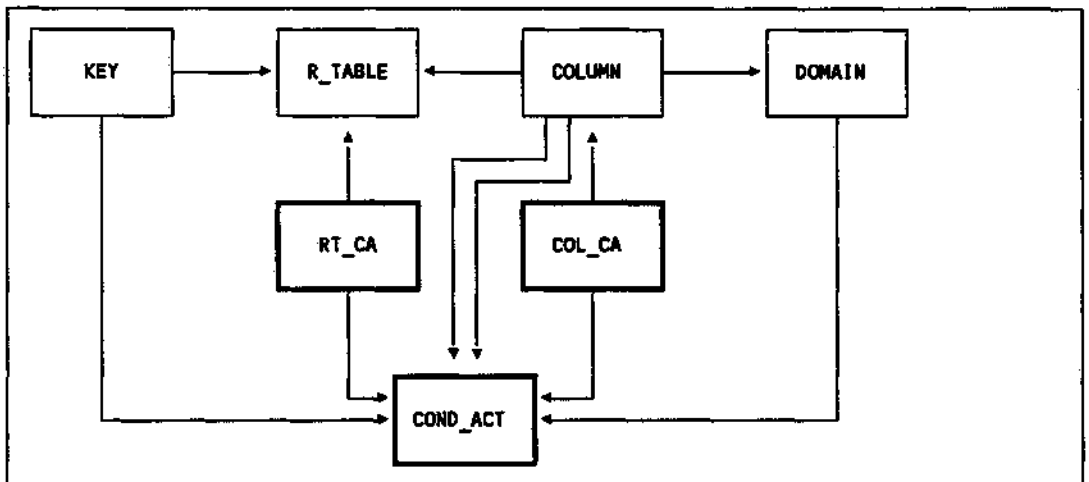


Figure 7a: Catalog support for constraints and triggered actions

A conditional action (COND_ACT) is either an integrity constraint or a clock-triggered action. Conditional actions are triggered by a statement in RL (CONDITION), in which case it is a constraint, or by the passing of time (CA_DATE, CA_TIME, INTERVAL), in which case it is a clock-triggered action. Constraints are enforced at the end of a request (timing type TC) or at the end of a transaction (timing type TT). Column TTIME specifies this timing

type. Column IND_STATIC specifies whether the constraint is a static constraint ('YES') or a transition constraint ('NO'). Only static constraints can be executed by the DBA as a command. It is assumed that the RL makes it possible to distinguish between the values of a data item before and after the change. The executable routine (TA_EXEC), triggered by the constraint or clock-triggered action is known to the catalog.

Codd distinguishes between entity integrity (E), referential integrity (R), column integrity (C), domain integrity (D), and user-defined integrity (U) constraints. The kind of constraint is determined by the tuples that reference a given COND_ACT-tuple. Every integrity constraint can be invoked as a command by means of its unique name (CA_NM). E-constraints, R-constraints, and alternate keys are represented in the catalog by KEY-tuples with KEY-type = 'PRIMARY', 'FOREIGN' or 'ALTERNATE'. Every such tuple references a COND_ACT-tuple by its column CA_NM.

Aside: Codd's definition of entity integrity contains two parts. The first part is the well known rule disallowing columns belonging to a primary key to accept any marks. The second part disallows columns belonging to a foreign key to accept I-marks. Apart from doubts one may have about the appropriateness of such an extension, it seems more appropriate for an RDBMS to enforce it as part of R-type constraint enforcement.

For every KEY-tuple for which KEY_TYPE = 'FOREIGN', the DBA must specify the action to be taken in case of an attempt to update (UPD_ACT) or delete (DEL_ACT) a primary key of a tuple in the R-table referenced by the foreign key. The RDBMS may cascade the update or delete to all other foreign key columns defined on the same domain, replace such foreign key columns with A-marks, or reject the attempted operation altogether if there exist references to the tuple to be updated or deleted.

Every domain can have one D-constraint limiting the pool of values permitted by its basic data type. Analogously, every column can have one C-constraint (CA_NM_C) further limiting the pool of values permitted by its domain. In addition, a separate constraint can be specified (CA_NM_M) disallowing the occurrence of marks of either type. Note that specifying this constraint for primary key columns is redundant. For reasons of consistency and productivity, the contents of CONDITION in COND_ACT are maintained by the DBMS for constraints of type E and R, for alternate keys, for constraints that prohibit the use of marks and for generalized constraints (see below).

For user-defined constraints it is possible to specify the events that trigger the constraint. If the event is the insertion, deletion or retrieval of a tuple in an R-table this can be specified in RT_CA. If the event is the update of a column in a tuple this can be specified in COL_CA.

Aside: Codd stipulates that encryption and decryption, if desired, are the responsibility of the DBA, not of the RDBMS [p.358]. Encryption and decryption can be easily supported by the catalog. The same observation holds for automatic archiving [p.357].

Contrary to his general position on the subject, Codd specifies that "*a fully relational DBMS should have the capability of storing (in its catalog) statements defining the various kinds of dependencies - including the functional, multi-valued, join and inclusion types of dependencies - as they apply to the database being managed*" [p.193]. Generalized support for these rarely encountered constraints requires a further extension of the

catalog. A column CONSTR_TYPE, designating the kind of constraint is added to COND_ACT and a column GROUP_ID is added to COL_CA.

Functional dependencies that are not a consequence of the primary key can be supported by adding a COND_ACT-tuple for which CONSTR_TYPE = 'FD'. The columns that participate in the functional dependency constraint are designated in COL_CA. The column GROUP_ID indicates whether the column participating in the functional dependency constraint is functionally dependent on other columns. For the FD "A1,A2,...,Am ---> B1,B2,...,Bn" the A-columns are assigned GROUP_ID = 'DETERMINING', while the B-columns are assigned GROUP_ID = 'DEPENDENT'.

Multi-valued dependencies can be supported by adding a COND_ACT-tuple for which CONSTR_TYPE = 'MVD'. The columns that participate in the multi-valued dependency constraint are designated in COL_CA. For the MVD "A1,A2,...,Am ---> B1,B2,...,Bn / C1,C2,...,Cp" the A-columns are assigned GROUP_ID = 'DETERMINING', the B-columns are assigned GROUP_ID = 'DEPENDENT', and the C-columns are assigned GROUP_ID = 'INDEPENDENT'.

Join dependencies can be supported by adding a COND_ACT-tuple for which CONSTR_TYPE = 'JD'. The columns that participate in the join dependency constraint are designated in COL_CA. For the JD "A1,A2,...,Am --> B1,B2,...,Bn * C1,C2,...,Cp * ...." the A-columns are assigned GROUP_ID = 'DETERMINING', while the other columns are assigned different values for GROUP_ID.

Inclusion dependencies that are not a consequence of a foreign key declaration can be supported by adding a COND_ACT-tuple for which CONSTR_TYPE = 'INCL_DPCY'. The columns that participate in the inclusion dependency constraint are designated in COL_CA. For the inclusion dependency "A1,A2,...,An is-in B1,B2,...,Bn" the A-columns are assigned GROUP_ID = 'SUB', while the other columns are assigned GROUP_ID = 'SUPER'. Every A-column must match a B-column belonging to the same domain. If several A- or B-columns belong to the same domain, the resulting ambiguity has to be resolved by means of the declaration of composite columns and domains.

A consequence of the support of RM/V2 for these types of generalized constraints is that the DBMS now knows in what normal form the R-Tables (whether base or derived) are. This means that the DBMS is aware of update anomalies and will reject update attempts resulting in an inconsistent database state. Moreover, if a conceptual schema is denormalized for performance reasons, the RDBMS knows what the equivalent, normalized schema would be[4]. This means that Codd's requirement for explicit support for these so called "*conceptual relations*" [p.320] is superfluous.

---

[4] As discussed in [VELD91a], either the RDBMS or a utility program could exploit this knowledge to renormalize the database and rewrite the application programs.

Figure 7b lists the constraints the catalog has to enforce.

CA01- A COND_ACT-tuple can be referenced by a tuple in KEY, DOMAIN, COLUMN, RT_CA or COL_CA if and only if the column CONDITION is not I-marked.
CA02- If a COND_ACT-tuple is referenced by a KEY-tuple, then it must not be referenced by a tuple in DOMAIN, COLUMN, RT_CA or COL_CA.
CA03- If a COND_ACT-tuple is referenced by a DOMAIN-tuple, then it must not be referenced by a tuple in KEY, COLUMN, RT_CA or COL_CA.
CA04- If a COND_ACT-tuple is referenced by a COLUMN-tuple by means of its column CA_NM_C, then it must not be referenced by a tuple in KEY, DOMAIN, COLUMN, RT_CA or COL_CA.
CA05- If a COND_ACT-tuple is referenced by a COLUMN-tuple by means of its column CA_NM_M, then it must not be referenced by a tuple in KEY, DOMAIN, COLUMN, RT_CA or COL_CA.
CA06- If a COND_ACT-tuple is referenced by one or more tuples in RT_CA or COL_CA, then it must not be referenced by a tuple in KEY, DOMAIN or COLUMN.
CA07- The column CA_NM_M in COLUMN must be A-marked if and only if both A_MARK_ALL and I_MARK_ALL contain the value 'YES'.
CA08- Preferably, no COND_ACT-tuple that is referenced by either a DOMAIN-tuple (D-constraint) or a COLUMN-tuple (C-constraint) has a value 'TT' for TTIME.
CA09- Preferably, no tuple in COND_ACT, for which CONDITION is not I-marked, references tuples in R_TABLE having RT_TYPE <> 'BASE' via RT_CA, COL_CA, COLUMN, or KEY having KEY_TYPE not equal to 'PRIMARY'.
CA10- If INTERVAL in COND_ACT does not contain an I-mark, CA_DATE and CA_TIME may not contain such a mark either.
CA11- If a COND_ACT-tuple designates a dynamic constraint (IND_STATIC = 'NO'), then the column CONDITION must not be I-marked, and the columns CA_DATE and CONSTR_TYPE must be I-marked,
CA12- The columns CA_DATE and CA_TIME in COND_ACT are either both I-marked or not marked.
CA13- The columns CONDITION and TTIME in COND_ACT are either both I-marked or not marked.
CA14- The column CA_NM in KEY must be A-marked if and only if KEY_TYPE = 'INDEX'.
CA15- Every COLUMN-tuple that belongs to a primary key has an I-mark for CA_NM_M.
CA16- The columns UPD_ACT and DEL_ACT in KEY must be I-marked if and only if KEY_TYPE <> 'FOREIGN'.
CA17- Tuples in KEY for which KEY_TYPE = 'PRIMARY' reference tuples in COND_ACT for which TTIME = 'TC'.
CA18- Tuples in COND_ACT for which CONSTR_TYPE is not I-marked cannot accept I-marks for CONDITION.
CA19- The column GROUP_ID in COL_CA must be I-marked if and only if the column CONSTR_TYPE in COND_ACT is also I-marked.

Figure 7b: User-defined constraints for constraints and triggered actions *(continued on the next page)*

CA20- Tuples in COND_ACT for which CONSTR_TYPE = 'FD' are referenced by at least one COL_CA-tuple for which GROUP_ID = 'DETERMINING', by at least one COL_CA-tuple with GROUP_ID = 'DEPENDENT', by no other tuples in COL_CA, by one tuple in RT_CA, and by no tuple in COLUMN, DOMAIN of KEY. The columns refer to the same R-table and the RT_CA-tuple has the value 'YES' values for the column INS 'No' values for the columns 'DEL' and 'RET'.

CA21- Tuples in COND_ACT for which CONSTR_TYPE = 'MVD' are referenced by at least one COL_CA-tuple for which GROUP_ID = 'DETERMINING', by at least one COL_CA-tuple with GROUP_ID = 'DEPENDENT', by at least one COL_CA-tuple with GROUP_ID = 'INDEPENDENT', by no other tuples in COL_CA, by one tuple in RT_CA, and by no tuple in COLUMN, DOMAIN of KEY. The columns refer to the same R-table and the RT_CA-tuple has the value 'YES' values for the column INS 'No' values for the columns 'DEL' and 'RET'.

CA22- Tuples in COND_ACT for which CONSTR_TYPE = 'JD' are referenced by at least one COL_CA-tuple for which GROUP_ID = 'DETERMINING', by at least one COL_CA-tuple with a different GROUP_ID, by no other tuples in COL_CA, by one tuple in RT_CA, and by no tuple in COLUMN, DOMAIN of KEY. The columns refer to the same R-table and the RT_CA-tuple has the value 'YES' values for the column INS 'No' values for the columns 'DEL' and 'RET'.

CA23- The set of COL_CA-tuples having the value 'DETERMINING' for the column GROUP_ID and referencing a COND_ACT-tuple for which CONSTR_TYPE = 'FD' does not include all primary key columns of the referenced R-table.

CA24- Tuples in COND_ACT for which CONSTR_TYPE = 'INCL_DPCY' are referenced by pairs of COL_CA-tuples for which GROUP_ID = 'SUB' and for which GROUP_ID = 'SUPER', belonging to the same domain.

CA25- Tuples in COL_CA, referring to a tuple in COND_ACT for which CONSTR_TYPE = 'INCL_DPCY' and having the same value for GROUP_ID, have the same values for RT_NM. The column GROUP_ID may only accept the values 'SUB' and 'SUPER'.

CA26- Columns in RT_CA referencing a COND_ACT tuple for which CONSTR_TYPE = 'INCL_DPCY' and referenced by COL_CA-tuples for which GROUP_ID = 'SUPER' have the value 'YES' for the column DEL and the value 'NO' for their columns INS and RET, while columns in RT_CA referencing a COND_ACT tuple for which CONSTR_TYPE = 'INCL_DPCY' and referenced by COL_CA-tuples for which GROUP_ID = 'SUB' have the value 'YES' for the column INS and the value 'NO' for their columns DEL and RET.

CA27- The set of tuples in COL_CA, referring to a tuple in COND_ACT for which CONSTR_TYPE = 'INCL_DPCY', and for which GROUP_ID = 'SUB', does not coincide with a set of KEY_COLUMN-tuples referring to a KEY-tuple for which KEY_TYPE = 'FOREIGN', while at the same time the corresponding set of tuples in COL_CA, for which GROUP_ID = 'SUPER', refers to a KEY-tuple for which KEY_TYPE = 'PRIMARY'.

Figure 7b: User-defined constraints for constraints and triggered actions (continued)

Due to the occurrence of generalization problems, the number of ad hoc constraints is huge and many of these constraints are quite complex. As discussed above, the only remedy is to extend the relational model further to deal with generalization structures. Most constraints require no further elaboration. Constraint CA24 cannot be fully supported by SQL (see appendix

B). In section 10 one extra constraint with respect to conditional actions will be introduced.

A final remark concerns Codd's demands that the sequence in which constraints are executed must not affect the outcome of a transaction and that constraints should not be inconsistent with each other. It is hard to see how future RDBMS products can comply with these demands with respect to non-generalized constraints. Codd himself is not very optimistic about the emergence of tools that "*simplify the discovery of inconsistencies between integrity constraints*" [p.266]. If his position with regard to constraint generalization prevails, he will probably prove right.

8. BUILT-IN AND USER-DEFINED FUNCTIONS

RM/V2 requires RDBMSs to support user-defined functions. These functions can be implemented by means of a host-language, supported by the RDBMS. Figure 8a shows the extensions to the catalog to support both built-in and user-defined functions.

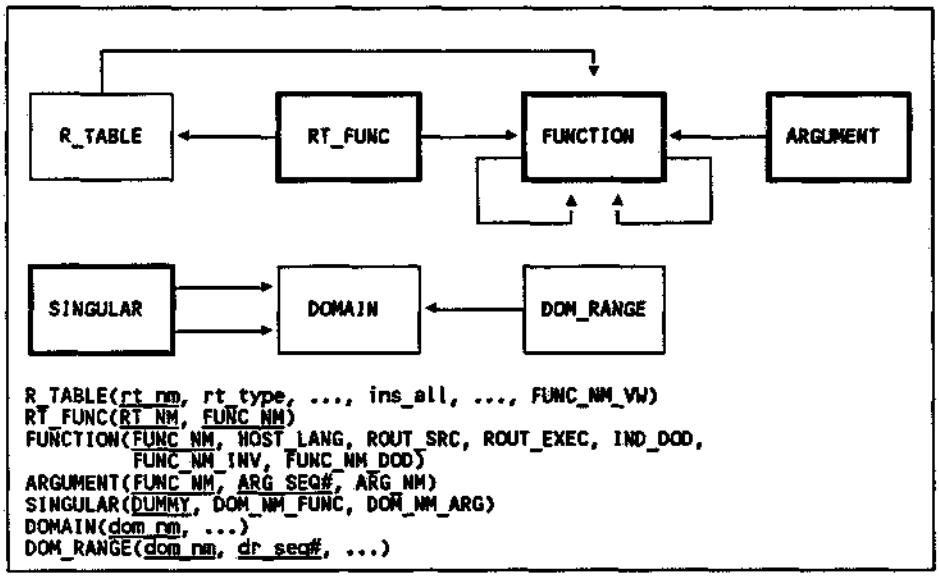


Figure 8a: Catalog support for functions

For every function, whether built-in or user-defined, the catalog contains a FUNCTION-tuple. A function is identified by its name (FUNC_NM). User-defined functions can be defined by means of a host-language (HOST_LANG), supported by the RDBMS. DBMS support for FORTRAN, COBOL and PL/1 is mandatory. The identifying names of both the source routine (ROUT_SRC) and the executable routine (ROUT_EXEC) is stored in the catalog. The column IND_DOD indicates for statistical functions whether the function takes into account the *degree of duplication*. The non-DOD version of a statistical function makes a projection of an R-table over those columns that constitute its arguments [p.340]. If a function has an inverse, this inverse is designated as FUNC_NM_INV. If a function has a (non-)DOD counterpart it is designated as FUNC_NM_DOD.

The relation R_TABLE has been extended with the column FUNC_NM_VW. This column makes it possible for the RDBMS to support inserts on views that are a union of several relations.

The relation RT_FUNC designates the R-tables to which a function must

have read-only access.
The relation ARGUMENT contains for every function its arguments. The columns FUNC_NM and ARG_NM constitute an alternate key.

Codd requires RM/V2 DBMSs to support two special kinds of domains, whose permitted values are determined by the existing functions and the existing arguments respectively. To avoid the hiding of information, a relation SINGULAR, containing the names of these special domains, has been added to the catalog. Because this relation must contain exactly one tuple, there does not exist a natural primary key (see [VELD91b]). Therefore, a dummy column is designated as the primary key.
Figure 8b lists the user-defined constraints necessary to provide adequate support for functions.

FU01- If FUNC_NM_INV is not A-marked, the FUNCTION-tuple it references contains the value of FUNC_NM for its column FUNC_NM_INV.
FU02- If FUNC_NM_DOD is not A-marked, the FUNCTION-tuple it references contains the value of FUNC_NM for its column FUNC_NM_DOD.
FU03- If FUNC_NM_DOD is not A-marked, then IND_DOD must not be I-marked.
FU04- Functions for which FUNC_NM_DOD is not I-marked reference FUNCTION-tuples for which IND_DOD has a different value.
FU05- R_TABLE-tuples for which RT_TYPE <> 'VIEW' or INS_ALL = 'NO' have A-marks for FUNC_NM_VW.
FU06- For every tuple in R_TABLE for which FUNC_NM_VW is not A-marked there exists a tuple in RT_FUNC with the same values for RT_NM and FUNC_NM.
FU07- FUNCTION-tuples that are each others inverse or that constitute a DOD/non-DOD pair are referenced by identical sets of ARGUMENT-tuples, projected on columns ARG_SEQ# and ARG_NM.
FU08- The values of ARG_SEQ# in the set of ARGUMENT-tuples with the same value for FUNC_NM are numbered consecutively up from 1 to the number of tuples in the set.
FU09- The relation SINGULAR contains exactly one tuple.
FU10- The values of columns DOM_NM_FUNC and DOM_NM_ARG in SINGULAR are not equal.

Figure 8b: Constraints for functions

Constraints FU06 and FU08 will be trivially modified and two constraints will be added in section 10 (see figures 10c and 10d).

9. SYNONYM NAMES AND NAMING RULES

RM/V2 provides a large number of rules concerning the naming of relational objects. Many of these rules have not been described explicitly, because they are enforced by the data structures and the constraints applying to these structures. There remain, however, a number of rules that have yet to be specified as user-defined constraints. Figure 9a (overleaf) contains the part of the catalog affected by the naming rules.

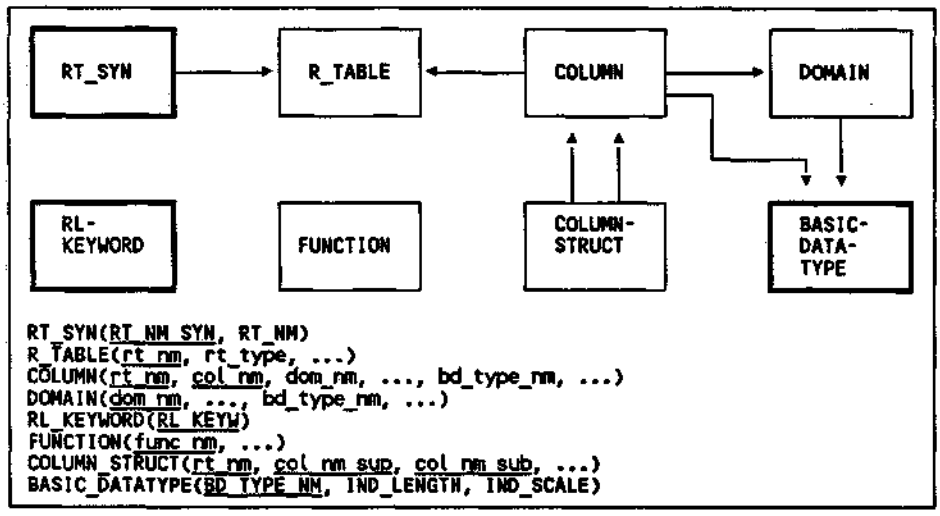


Figure 9a: Catalog support for synonyms and naming rules

Because it must be possible to assign alternate names to base R-tables and views, the relation RT_SYN is introduced. The unary relation RL_KEYWORD is added because names of relational objects must be different from reserved words of the relational language. Because some naming rules apply to all data types, whether extended (domains) or basic, the relation BASIC_DATATYPE is introduced. The columns IND_LENGTH and IND_SCALE designate whether the columns LENGTH and SCALE in COLUMN and DOMAIN are relevant for the basic data type. Like RL_KEYWORD, BASIC_DATATYPE is RDBMS-supplied and not updatable by a user. These relations make it possible to define all naming rules without having to resort to enumerating arbitrarily large numbers of basic data types and language keywords relevant to the particular RDBMS.
Figure 9b lists the constraints that enforce the additional naming rules.

NR01- The sets of R-table names designated by columns RT_NM_SYN and RT_NM in RT_SYN are disjunct.

NR02- The set of composite column names designated by COL_NM_SUP in COL_STRUCT is disjunct from the set of keywords in the relation RL_KEYWORD.

NR03- The set of domain names designated by DOM_NM in DOMAIN is disjunct from the union of RT_NM_SYN in RT_SYN and RT_NM in R_TABLE.

NR04- The sets of names designated by RT_NM_SYN in RT_SYN, RT_NM in R_TABLE and FUNC_NM in FUNCTION are distinct from another and from the union of DOM_NM in DOMAIN, BD_TYPE_NM in BASIC_DATATYPE, and COL_NM in COLUMN.

NR05- Every DOMAIN-tuple and every COLUMN-tuple, that references a BASIC_DATATYPE-tuple for which IND_LENGTH = 'YES', does not have a marked value for its LENGTH column, while the LENGTH column is I-marked if IND_LENGTH = 'NO'.

NR06- Every DOMAIN-tuple and every COLUMN-tuple, that references a BASIC_DATATYPE-tuple for which IND_SCALE = 'YES', does not have a marked value for its SCALE column, while the SCALE column is I-marked if IND_SCALE = 'NO'.

Figure 9b: Constraints for naming rule enforcement

Constraints NR01 and NR04 will be trivially modified in the next section

(see figure 10c).

Figure 9c provides an overview of naming restrictions for every combination of relational objects. It appears that there are a number of combinations for which Codd does not specify naming rules. For most of these combinations the authors feel that the sets of names should be disjunct.

	R-tables	Columns	Bas. Datatype	Domains	Functions
Columns	not allowed				
Basic Datatypes	not allowed	?			
Domains	not allowed	allowed	?		
Functions	allowed	allowed	not allowed	not allowed	
Keywords	?	not allowed	?	?	?

Figure 9c: Permissibility of duplicate names between relational object classes

A last observation concerns the fact that expression of constraints NR02 to NR04 requires comparing columns that belong to different domains. Thus, to enforce these constraints a *domain check override* [p.238] must be specified.

10. DATABASE USERS

So far the multi-user nature of an RM/V2-DBMS has been disregarded, but at this point the database user is introduced as a separate entity, in order to pave the way for a discussion in the next section of the numerous RM/V2 authorization features. This section is concerned with the incorporation of user-references in the existing catalog structure.

The main question is which relational objects are user-dependent and which are not. The principal relational objects are R-tables, domains, functions, and conditional actions (constraints and clock-triggered actions). Unfortunately, Codd provides no information about how to deal with these objects with regard to database users. We have chosen to consider only domains to be user-independent. This means that in a database only domains are uniquely defined by means of their name (DOM_NM). R-tables are identified by their owner's identification and their table name. The same holds for functions and conditional actions. Thus, a user-identification has to be added to the primary keys of the catalog tables representing these three objects, and this change in keys is cascaded to all tables that reference them. Figure 10a (overleaf) lists the catalog tables to which one or more columns belonging to the domain USER_NM have been added. The tables are listed in order of their introduction in the preceding sections.

Catalog Table	Change in primary key	Change in foreign key-non primary key
R_TABLE	USER_NM, rt_nm	
COLUMN	USER_NM, rt_nm, col_nm	USER_NM_BASE, rt_nm_base, col_nm_base
DOMAIN		USER_NM_DOM and USER_NM_CA, ca_nm
KEY_REF	USER_NM_PK, rt_nm_pk, USER_NM_FK, rt_nm_fk, key_nm	
KEY	USER_NM, rt_nm, key_nm	
KEY_COLUMN	USER_NM, rt_nm, key_nm, col_nm	
COLUMN_STRUCT	USER_NM, rt_nm, col_nm_sup, col_nm_sub	
ARCHIVE	USER_NM, archive_nm	
DB_INDEX_COL	USER_NM_DBI, index_nm, USER_NM_RT, rt_nm, col_nm	
DB_INDEX	USER_NM, index_nm	
RT_CA	USER_NM_RT, rt_nm, USER_NM_CA, ca_nm	
COL_CA	USER_NM_RT, rt_nm, USER_NM_CA, col_nm, ca_nm	
COND_ACT	USER_NM, ca_nm	
RT_FUNC	USER_NM_RT, rt_nm, USER_NM_FUNC, func_nm	
FUNCTION	USER_NM, func_nm	
ARGUMENT	USER_NM, func_nm, arg_seq#	
SINGULAR		USER_NM_CATALOG
RT_SYN	USER_NM_SYN, rt_nm_syn, USER_NM_RT	USER_NM_RT, rt_nm

Figure 10a: Changes in catalog table structures due to the addition database users.

Only columns constituting primary keys and foreign key references have been changed. The references themselves, depicted by arrows in the preceding figures, remain unchanged. However, the newly introduced USER_NM column refers to a relation USER that contains the identification of the RDBMS users. Figure 10b shows this extension to the catalog.

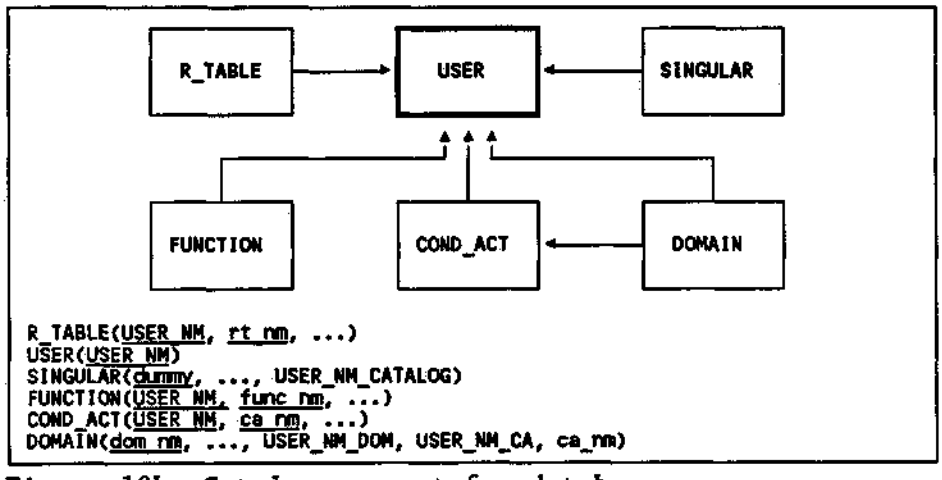


Figure 10b: Catalog support for database users

Figures 10a and 10b show that, as far as identification is concerned, R-tables, functions and conditional actions are completely local to a user. As a consequence, the foreign keys of all catalog relations referencing these three relations have been changed too. Domains remain uniquely identified by the column DOM_NM. The addition of the columns USER_NM_DOM and USER_NM_CA only serves to identify the user who has created the domain and the user to whom a possible D-constraint belongs. There exists one system defined user who owns the catalog database. This user is identified by the column USER_NM_CATALOG in the relation SINGULAR.

Except for domains, the adopted changes give database users much freedom in naming their objects. Users can assign names to R-tables, columns, functions, constraints, and so on without taking the names of objects

created by others into account. We believe that not making domains user-dependent is in line with the philosophy behind these objects [Ch.3]. For R-tables, a user-dependent naming scheme is customary in current RDBMS products. A similar scheme is adopted for user-defined functions and conditional actions.

Generally, database objects like R-tables, columns, functions, etcetera. that are related to each other are owned by the same user. However, the catalog permits its users to exploit each others objects in a number of cases. Where this is the case, *two* columns belonging to the domain of user-identifications have been added. Obviously, these columns need not contain the same value. Such different values are only allowed if permitted by the authorization data in the catalog (see next section). Figure 10a shows that multiple user-identifications exist in the relations KEY_REF, COLUMN, DOMAIN, DB_INDEX_COL, RT_CA, COL_CA, RT_FUNC and RT_SYN.

Multiple user-identifications in KEY_REF enable the user to integrate his database schema with another user's schema, which can be particularly useful if a database is integrated with the catalog database. Multiple user-identifications in COLUMN enable the view owner to reference a source column that belongs to a relation owned by another user. Multiple user-identifications in DB_INDEX_COL enable users to incorporate columns belonging to relations created by different users in one domain-based index. Multiple user-identifications in RT_FUNC, RT_CA and COL_CA enable the owner of the conditional action or user-defined function to refer to other users' R-tables and columns respectively. Finally, the changed structure of RT_SYN permits the user to create a synonym name for an R-table owned by another user.

Figure 10c revisits a number of constraints that were defined in previous sections, but have to be trivially modified due to the changes introduced in this section. The modifications are printed in italic format.

BM04- The subset of KEY-tuples for which KEY_TYPE = 'PRIMARY' does not contain duplicate values for *USER_NM and* RT_NM.

BM11- The values of COL_SEQ# in the set of COLUMN-tuples with the same value for *USER_NM and* RT_NM are numbered consecutively up from 1 to the number of tuples in the set.

BM12- The values of KEY_COL_SEQ# in the set of KEY_COLUMN-tuples with the same value for *USER_NM,* RT_NM and KEY_NM are numbered consecutively up from 1 to the number of tuples in the set.

CM01- *For ever value of USER_NM,* the set of tuples in COL_NM_SUP, projected over USER_NM, RT_NM and COL_NM_SUP, is disjunct from the set of tuples projected over USER_NM, RT_NM and COL_NM_SUB.

CM03- The values of CS_SEQ# in the set of COLUMN_STRUCT-tuples with the same value for *USER_NM and* COL_NM_SUP are numbered consecutively up from 1 to the number of tuples in the set.

VW02- For every COLUMN-tuple the columns *USER_NM,* RT_NM_BASE and COL_NM_BASE are either I-marked or not marked.

IX01- The projections of DB_INDEX and KEY on their columns *USER_NM and* INDEX_NM are disjunct.

Figure 10c: Constraints modified due to the introduction of users
(continued on the next page)

IX04- The projection of the columns *USER_NM_RT*, RT_NM and COL_NM in DB_INDEX_COL for one given value of INDEX_NM is not a subset of another such projection for another value of INDEX_NM.

IX08- For the subset of KEY-tuples for which INDEX_NM is not I-marked, the projection of the columns *USER_NM*, RT_NM, COL_NM and KEY_COL_SEQ# in KEY_COLUMN for one given value of KEY_NM is not a subset of another such projection for another KEY-tuple.

FU06- For every tuple in R_TABLE for which FUNC_NM_VW is not A-marked there exists a tuple in RT_FUNC with the same values for RT_NM and FUNC_NM *and with values for USER_NM_RT and USER_NM_FUNC that are equal to USER_NM in R_TABLE*.

FU08- The values of ARG_SEQ# in the set of ARGUMENT-tuples with the same value for *USER_NM and* FUNC_NM are numbered consecutively up from 1 to the number of tuples in the set.

NR01- The sets of R_TABLE-tuples designated by columns *USER_NM_SYN and* RT_NM_SYN *on the one hand* and *by USER_NM_RT and* RT_NM in RT_SYN are disjunct.

NR04- *For every user*, the sets of names designated by RT_NM_SYN in RT_SYN, RT_NM in R_TABLE and FUNC_NM in FUNCTION are distinct from another and from the union of DOM_NM in DOMAIN, BD_TYPE_NM in BASIC_DATATYPE, and COL_NM in COLUMN *for that user*.

Figure 10c: Constraints modified due to the introduction of users (continued)

In discussing domains and functions, the fact that the catalog must be able to distinguish between built-in and user-defined domains and functions has been ignored. The extension of the catalog with users and the designation of a special user who owns the catalog database itself make this distinction possible. Domains and functions owned by the catalog user are built-in, other domains and functions are not. This makes it possible to add two constraints that could not be expressed in section 8 (see figure 10d).

Finally, figure 10a shows that the introduction of users changes a simple foreign key into a composite one. Due to Codd's very liberal definition of referential integrity [p.23], which among others allows composite foreign keys to be partly A-marked, an ad hoc constraint has to be introduced to forbid such situations from occurring.

Figure 10d lists the extra constraints discussed above.

CA28- For every DOMAIN-tuple the columns USER_NM_CA and CA_NM are either I-marked or not marked.

FU11- If the USER_NM in FUNCTION is not equal to USER_NM_CATALOG in SINGULAR, the columns HOST_LANG, ROUT_SRC and ROUT_EXEC must not be marked.

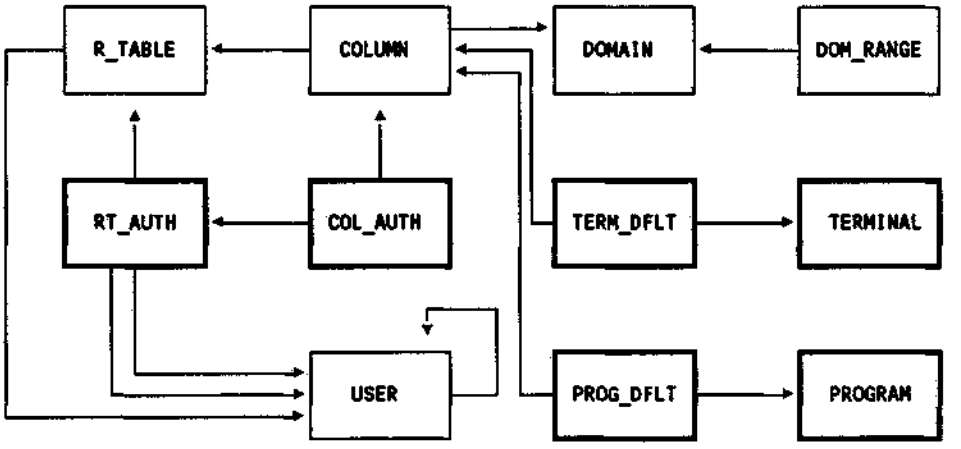
FU12- Every DOMAIN-tuple that is referenced from SINGULAR has values for USER_NM_DOM and USER_NM_CA that are equal to USER_NM_CATALOG in SINGULAR.

Figure 10d: Constraints added due to the introduction of users

11. AUTHORIZATION

The information concerning authorization is of course to be found in the catalog and is thus accessible to any sufficiently authorized user. However, decisions must be made about the extent to which the catalog should support authorization by means of specific data structures, in addition to support for free-format RL-expressions. This problem was amply discussed in section 7 with respect to constraint specification, and will not be revisited here. We have chosen to provide catalog data structures for *user* authorization for R-tables and columns, because terminal users and application programs will frequently query this information and because this type of authorization is also supported by catalogs of existing RDBMS-products. *Terminal* and *application program* authorization constraints with respect to R-tables and columns are supported by RL-expressions only. The same holds for authorization constraints on system dates and times. To allow the expression of authorization constraints per terminal or per application program, relations containing these objects are added to the catalog.

Figure 11a shows the extensions to the catalog necessary to support Codd's specifications concerning authorization.




```
R_TABLE(user_nm, rt_nm, ..., N_DROP, N_DELETE, N_DELAY, DELAY_PERIOD)
COLUMN(user_nm, rt_nm, col_nm, dom_nm, a_mark_all, i_mark_all, ..., MARK_PREF)
DOMAIN(dom_nm, ..., bd_type_nm, length, scale, ...)
DOM_RANGE(dom_nm, dr_seq#, loval, hival)
RT_AUTH(USER_NM_RT, RT_NM, USER_NM_GRANTOR, USER_NM_GRANTEE, RET_ALL, INS_ALL, PK_UPD_ALL, ARCH_ALL,
        DEL_ALL, SNAPSHOT_ALL, CHECKOUT_ALL, PK_HIDDEN, RL_EXPR_RET, RL_EXPR_INS, RL_EXPR_PK_UPD,
        RL_EXPR_ARCH, RL_EXPR_DEL, RL_EXPR_SNAPSHOT, RL_EXPR_CHECKOUT)
COL_AUTH(USER_NM_RT, RT_NM, COL_NM, USER_NM_GRANTOR, USER_NM_GRANTEE, UPD_ALL, RL_EXPR_UPD,
         DEFAULT_VAL, I_MARK_ALL, I_UNMARK_ALL,
TERM_DFLT(TERM_ID, USER_NM, RT_NM, COL_NM, DEFAULT_VAL)
TERMINAL(TERM_ID)
USER(user_nm, USER_GRP_NM, TIME_BLOCK, LOCK_QUOTA, MAX_STOR_QUERY_ANT, MAX_STOR_QUERY_ACT,
          MAX_STOR_TOTAL_ANT, MAX_STOR_TOTAL_ACT, DBC_DOM, DBC_RT, DBC_COL, DBC_VIEW, DBC_CA_E,
          DBC_CA_R, DBC_CA_C, DBC_CA_D, DBC_CA_U, DBC_FUNC, DBC_ACCESS, DBC_DB_INDEX, DBC_FK,
          DBC_AUTH, DBC_SNAPSHOT, DBC_AUDIT_LOG, DBC_ARCH, DBC_DATE_ROUND, Q_A_MAYBE_ALL,
          Q_I_MAYBE_ALL, Q_MAYBE_ALL, Q_AR_ALL, Q_IR_ALL, Q_ESR_ALL, Q_ORDER_ALL, Q_ONCE_ALL,
          Q_DCO_ALL, Q_EXCL_SIBL_ALL, Q_DOD_ALL, Q_SAVE_ALL, Q_VALUE_ALL, IND_DBA_USER, GRANT_ALL,
          ASSIGN_ALL)
PROG_DFLT(PROG_ID, USER_NM, RT_NM, COL_NM, DEFAULT_VAL)
PROGRAM(PROG_ID, TIME_BLOCK, LOCK_QUOTA)
```

Figure 11a: Catalog support for authorization

With respect to authorization the core catalog entity is the relation USER, as discussed in the previous section. To support authorization, a large number of columns must be added to this relation. Codd requires that, in

order to make the task of the DBA easier, authorization must be possible on the *user group* level in addition to the individual user level [p.334]. To support this a column USER_GRP_NM is added to the relation USER. Because this column is a foreign key to the USER relation itself, USER-tuples now represent either individual users or user groups. An individual user can either be specifically authorized for some action or inherit his authorization from his group. Individual authorization overrides group authorization.

Users' access to R-tables and columns is implemented by means of the relations RT_AUTH and COL_AUTH. These relations may contain references to three distinct users. The column USER_NM_RT refers to the user who has created the R-table. The columns USER_NM_GRANTOR and USER_NM_GRANTEE refer to the authorizing and the authorized user. This scheme permits the catalog to reflect not only the authorizations in existence, but also their sources. This information is needed whenever an authorization is revoked, because such an action may cascade to other authorizations. Moreover, revoking an authorization should not affect a user's privileges if he or she has been authorized from another source [p.334]. The result of the emphasis on user groups and authorization sources is that a user's actual authorizations cannot be retrieved by a simple query. The relation RT_AUTH contains information per R_Table:

1- whether retrieval is allowed (RET_ALL);
2- whether insertion is allowed (INS_ALL);
3- whether update of primary key columns is allowed (PK_UPD_ALL);
4- whether archiving and restoring is allowed (ARCH_ALL);
5- whether deletion is allowed (DEL_ALL);
6- whether snapshot creation is allowed (SNAPSHOT_ALL);
7- whether checkout creation is allowed (CHECKOUT_ALL);
8- whether the values of the primary key columns should be hidden to the authorized user (PK_HIDDEN), in order to give him or her the opportunity to query the R_Table for statistical purposes [p.329].

Besides these standard authorizations, the catalog makes it possible to constrain users' authorizations further by means of expressions in RL. For instance, these expressions make it possible to limit access to working hours or to specific terminals. Columns RL_EXPR_RET, RL_EXPR_INS, RL_EXPR_PK_UPD, RL_EXPR_ARCH, RL_EXPR_DEL, RL_EXPR_SNAPSHOT, and RL_EXPR_CHECKOUT contain these expressions. The authorization to update a column is expressed in the relation COL_AUTH by means of the columns UPD_ALL and RL_EXPR_UPD.

Three columns are added to the relation R_TABLE to support Codd's N-person Turn-key feature (RA-5). Each column contains the minimal number of authorized users who must perform the commands 'DROP RELATION' (N_DROP), 'DELETE' (N_DELETE), or the command that changes the delay-period (DELAY_PERIOD) after which the RDBMS effects a 'DROP RELATION' command (N_DELAY).

The column MARK_PREF is added to the relation COLUMN to allow the authorized user to specify a mark preference [p.267/268] in case:

1- neither a value nor a mark is specified for the column at insert time;
2- both types of marks are allowed;
3- there does not exist a default value for the column (see below);
4- there does not exist an integrity constraint that generates a mark.

Default values can be specified per terminal or per application program [p.267]. To support this, the relations TERM_DFLT and PROG_DFLT, containing these defaults (DEFAULT_VAL), are added to the catalog. Although Codd makes no mention of it, a similar default per user is included in the catalog, by means of the column DEFAULT_VAL in COL_AUTH.

To avoid long-term locking, the DBA can specify a time block on every locking action caused by any request from a user or a program [p.353]. In addition, the DBA can specify a locking quota expressed as a number of time blocks. Catalog support for this feature is offered by the columns TIME_BLOCK and LOCK_QUOTA in relations USER and PROGRAM. If a locking request comes from a terminal user, the RDBMS inspects the USER relation, while it inspects the PROGRAM relation if the request is originated by an application program.

The relation USER contains a large number of columns with information about other authorizations. It has been assumed that it is not necessary to know which specific DBA-user has granted the authorizations.

The columns MAX_STOR_QUERY_ANT, MAX_STOR_QUERY_ACT, MAX_STOR_TOTAL_ANT and MAX_STOR_TOTAL_ACT contain maximum anticipated and actual storage parameters. Codd specifies that it must be possible to "*block execution of any query for which the result exceeds the quota of storage assigned to a user or a group of users, either on a per-query basis or with respect to a specified total*" [p.331].

In feature RA-7, Codd enumerates 13 database-control activities that are subject to authorization [p.331]. For each activity an indicator has been added to the USER relation. The database-control activities are:

1- creating and dropping a domain (DBC_DOM);
2- creating and dropping a base R-table (DBC_RT);
3- creating and dropping a column of an existing R-table (DBC_COL);
4- creating and dropping a view (DBC_VIEW);
5- creating and dropping an integrity constraint by type (DBC_CA_E, DBC_CA_R, DBC_CA_C, DBC_CA_D and DBC_CA_U);
6- creating and dropping a user-defined function (DBC_FUNC);
7- creating and dropping a performance-oriented access path like an index (DBC_ACCESS);
8- creating a foreign key referencing a primary key in some R-table (DBC_FK);
9- requesting that a specific authorization be granted or discontinued (DBC_AUTH);
10- requesting a snapshot (DBC_SNAPSHOT);
11- requesting that an audit log be maintained or discontinued (DBC_AUDIT_LOG);
12- establishing a condition for archiving with a new label (DBC_ARCH);
13- establishing an UP or DOWN mode for "rounding" pseudo-dates like 'February 30' (DBC_DATE_ROUND).

In feature RA-9, Codd enumerates 13 qualifiers the use of which is subject to authorization [p.333]. For each qualifier an indicator has been added to the USER relation. The qualifiers are:

1- A-MAYBE (Q_A_MAYBE_ALL);
2- I-MAYBE (Q_I_MAYBE_ALL);
3- MAYBE (Q_MAYBE_ALL);
4- AR(x): temporary replacement of A-marks (Q_AR_ALL);
5- IR(x): temporary replacement of I-marks (Q_IR_ALL);
6- ESR(x): temporary replacement of empty relations (Q_ESR_ALL);
7- ORDER BY (Q_ORDER_ALL);
8- ONCE ONLY (Q_ONCE_ALL);
9- DOMAIN CHECK OVERRIDE (Q_DCO_ALL);
10- EXCLUDE SIBLINGS (Q_EXCL_SIBL_ALL);
11- DEGREE OF DUPLICATION (Q_DOD_ALL);
12- SAVE (Q_SAVE_ALL);
13- VALUE: default value for newly created columns (Q_VALUE_ALL).

Finally, several columns are added to the USER relation expressing whether the user has DBA status (IND_DBA_USER), whether the user is allowed to pass on authorizations received (GRANT_ALL), and whether the user is allowed to grant authorizations to other users for which he or she is not authorized (ASSIGN_ALL).

Figure 11b lists the constraints with respect to authorization.

AU01-	The values of the columns N_DROP, N_DELETE and N_DELAY in R_TABLE must be less than the number of individual users in the relation USER.
AU02-	A COLUMN-tuple contains an I-MARK for MARK_PREF if and only if A_MARK_ALL and I_MARK_ALL both contain the value 'YES'.
AU03-	No RT_AUTH-tuple has the same values for USER_NM_RT and USER_NM_GRANTEE or USER_NM_GRANTOR and USER_NM_GRANTEE.
AU04-	Authorization cycles may not occur in RT_AUTH.
AU05-	For every RT_AUTH-tuple, a column containing an RL-expression must contain an I-mark if the corresponding indicator contains the value 'NO'.
AU06-	Every RT_AUTH-tuple for which RET_ALL = 'NO' has the same value for SNAPSHOT_ALL.
AU07-	For every COL_AUTH-tuple, the column RL_EXPR_UPD must contain an I-mark if the corresponding indicator UPD_ALL contains the value 'NO'.
AU08-	The column DEFAULT_VAL in COL_AUTH must conform to the domain specification of the domain to which the referenced COLUMN-tuple belongs.
AU09-	The column DEFAULT_VAL in TERM_DFLT must conform to the domain specification of the domain to which the referenced COLUMN-tuple belongs.
AU10-	The column DEFAULT_VAL in PROG_AUTH must conform to the domain specification of the domain to which the referenced COLUMN-tuple belongs.
AU11-	Every USER-tuple that is referenced by another USER-tuple has an A-mark for USER_GRP_NM.
AU12-	For no USER-tuple the storage authorizations per query (MAX_STOR_QUERY_ANT, MAX_STOR_QUERY_ACT) exceed their total counterparts (MAX_STOR_TOTAL_ANT, MAX_STOR_TOTAL_ACT).
AU13-	The column PK_HIDDEN in RT_AUTH must not contain an I-mark if and only if the referenced R-table has a primary key.
AU14-	The column CHECKOUT_ALL in RT_AUTH must not contain an I-mark if and only if the referenced R-table is a base R-table.
AU15-	Every USER-tuple for which IND_DBA_USER = 'YES' (DBA) has the value 'YES' for GRANT_ALL.

Figure 11b: Constraints for authorization

Constraints AU04, AU08, AU09 and AU10 cannot be expressed in RL and have to be supported by means of a lower level host-language. Constraint AU11 prohibits groups of user groups. Although there is no principal reason for this limitation, it is consistent to adopt the same restrictions with respect to user groups as we did in section 3 with respect to composite columns and composite domains.

Finally, we remark that the relation USER will be split in two relations in section 13, due to the introduction of database sites. This will result

in trivial modifications of constraints AU01, AU12 and AU15.

12. AUDIT LOGGING

Up to this point, the catalog model presented has been an example of a 'snapshot' database in which historic information plays no part. This may seem strange because database users (especially DBA users) and the RDBMS itself could make good use of catalog information concerning the evolution of a database schema. The lack of support for historic information applies only to the schema level, on the database level RM/V2 requires RDBMSs to support audit logging.

> *Aside*: Codd's specifications concerning audit logging require the RDBMS to support logging on the catalog database. If the RDBMS would require audit logging on catalog R-tables, the catalog would become a historic database.

Audit logging introduces the concept of *time*. As we saw in section 7 with respect to generalization, the Relational Model is unable to deal with time in an elegant manner (see [TSIC82] and [SNOD87]). On the modelling level we face the problem that we want the logged data to contain the primary key of the altered tuple, which consists of a variable number of columns. There exist two conventional approaches to the problem:

1- create one R-table for every R-table to be logged;
2- create one R-table with one column containing the R-table name and the concatenation of the primary key columns of the altered tuple.

The first approach would lead to an unwieldy catalog whose structure would change with the structure of the database. This would be unacceptable, especially in environments where alterations of R-table structures occur frequently. The second approach would violate the atomicity principle with respect to database values [p.6]. Codd does not provide any clues with respect to audit logging.

> *Aside:* Codd encounters a similar problem when he discusses a command whose function is to list all values drawn from a domain that exist in the database, regardless of the R-table in which they occur [p.56]. His suggestion is to introduce a *composite* column containing the primary keys of the various R-tables. The composite column should consist of a number of simple columns large enough to contain the largest primary key. This solution can be seen as an amalgamation of the two described approaches.

The solution we prefer is based on the concept of *partial implosion*, as introduced in [VELD91a] and [BOOG91], in which the distinction between data and metadata is partly blurred[5]. In this case the variable number of primary key columns is subjected to normalization. Figure 12a shows the resulting extension to the catalog.

---

[5] [VELD90] shows that it is possible to reduce any database schema to exactly *one*, fully *normalized relation* (full implosion). In [BOOG91a] this concept is applied to generalization hierarchies (partial implosion).

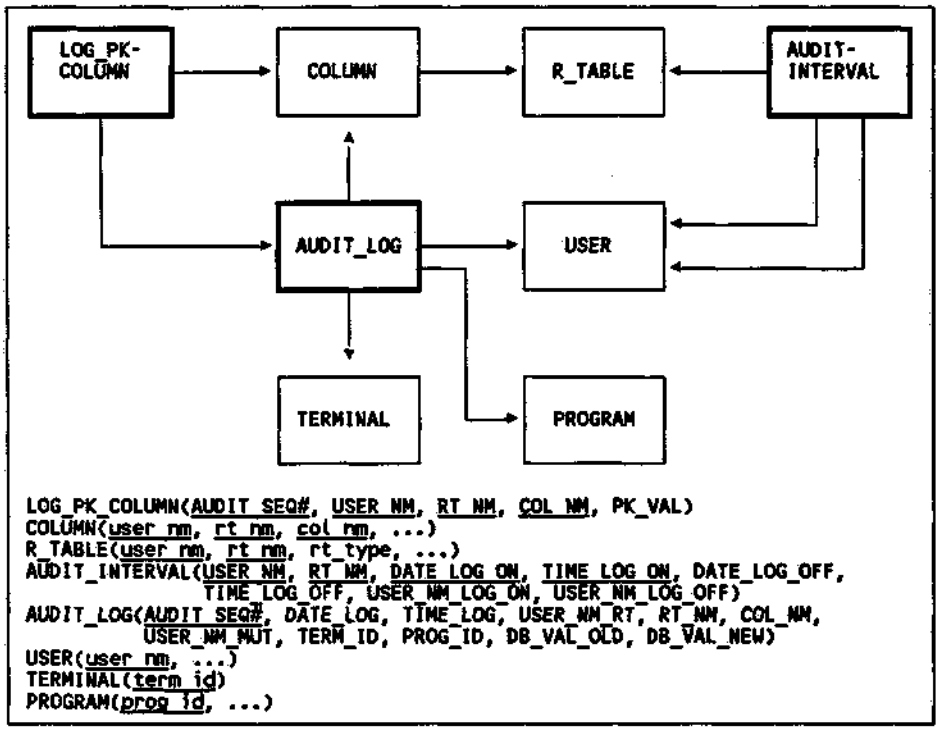


Figure 12a: Catalog support for audit logging

The relation AUDIT_LOG contains the logging data required by Codd. These include a time stamp (DATE_LOG, TIME_LOG), the identification of the altered column (USER_NM_RT, RT_NM, COL_NM), the identification of the user who has effected the database change (USER_NM_MUT), the identification of the terminal (TERM_ID) and the application program (PROG_ID), and the old and new values of the altered data item (DB_VAL_OLD, DB_VAL_NEW). Tuples in AUDIT_LOG are assigned a unique sequence number by the RDBMS (AUDIT_SEQ#).

The altered tuple is of course referred to by the value(s) of its primary key column(s). Because the number of columns is not fixed, the relation LOG_PK_COLUMN is introduced. For insertion in AUDIT_LOG, a number of tuples is inserted into LOG_PK_COLUMN, one for every primary key column of the altered R-table. A LOG_PK_COLUMN-tuple refers to an AUDIT_LOG-tuple (AUDIT_SEQ#) and to a tupel in COLUMN (USER_NM, RT_NM, COL_NM). The value of the primary key column is represented in the column PK_VAL.

We assume that audit logging can be specified per base R-table. We assume further that the catalog should contain the information about the audit logging actions performed by the RDBMS. Therefore, a relation AUDIT_INTERVAL, referencing the relation R_TABLE, is added to the catalog. For every logging interval on an R-table, the RDBMS inserts one tuple containing the R-Table's identification (USER_NM, RT_NM), timestamps for the moments audit logging is turned on (DATE_LOG_ON, TIME_LOG_ON) and off (DATE_LOG_OFF, TIME_LOG_OFF), and the identification for the user who turned audit logging on (USER_NM_LOG_ON) and off (USER_NM_LOG_OFF).

Note that the proposed catalog extension may cause problems if an attempt is made to delete a tuple in USER, COLUMN, TERMINAL, or PROGRAM. We assume here that the RDBMS software will accept the deletion of a column and cascade this deletion to referencing tuples in AUDIT_LOG and LOG_PK_COLUMN, while an attempt to delete a tuple in the other tables is rejected by the RDBMS software if it is referenced by one or more tuples in AUDIT_LOG.

Figure 12b lists the constraints with respect to audit logging.

LG01-	Every tuple in AUDIT_LOG references a tuple in COLUMN that references a tuple in R_TABLE for which RT_TYPE = 'BASE'
LG02-	Every tuple in AUDIT_LOG is referenced by at least one tuple in LOG_PK_COLUMN.
LG03-	The set of COLUMN-tuples referenced by tuples in LOG_PK_COLUMN and AUDIT_LOG, for which AUDIT_SEQ# is the same, all reference the same R_TABLE-tuple.
LG04-	For every tuple in AUDIT_INTERVAL the columns DATE_LOG_OFF, TIME_LOG_OFF and USER_NM_LOG_OFF are all either marked or unmarked.

Figure 12b: Constraints for audit logging

The number of constraints would be much larger if the catalog would contain historical information about schema evolution. Because the database schema may have been altered after the insertion of the logged data, the ability of the catalog to guard its integrity is degraded.

13. DISTRIBUTED DATABASE

The trend towards dirtributed database management is a major argument in favour of a vendor-independent catalog standard. Nevertheless, designing a catalog model that adequately supports this aspect is not easy, if only because a data distribution problem implies a catalog distribution problem. Just as data distribution leads to global and local databases, it leads to global and local catalogs. The additions to the catalog model presented in this section turn the catalog into a *global* catalog, from which local catalogs can be formally derived. Figures 13a (overleaf) and 13c show the proposed extensions to the catalog model.

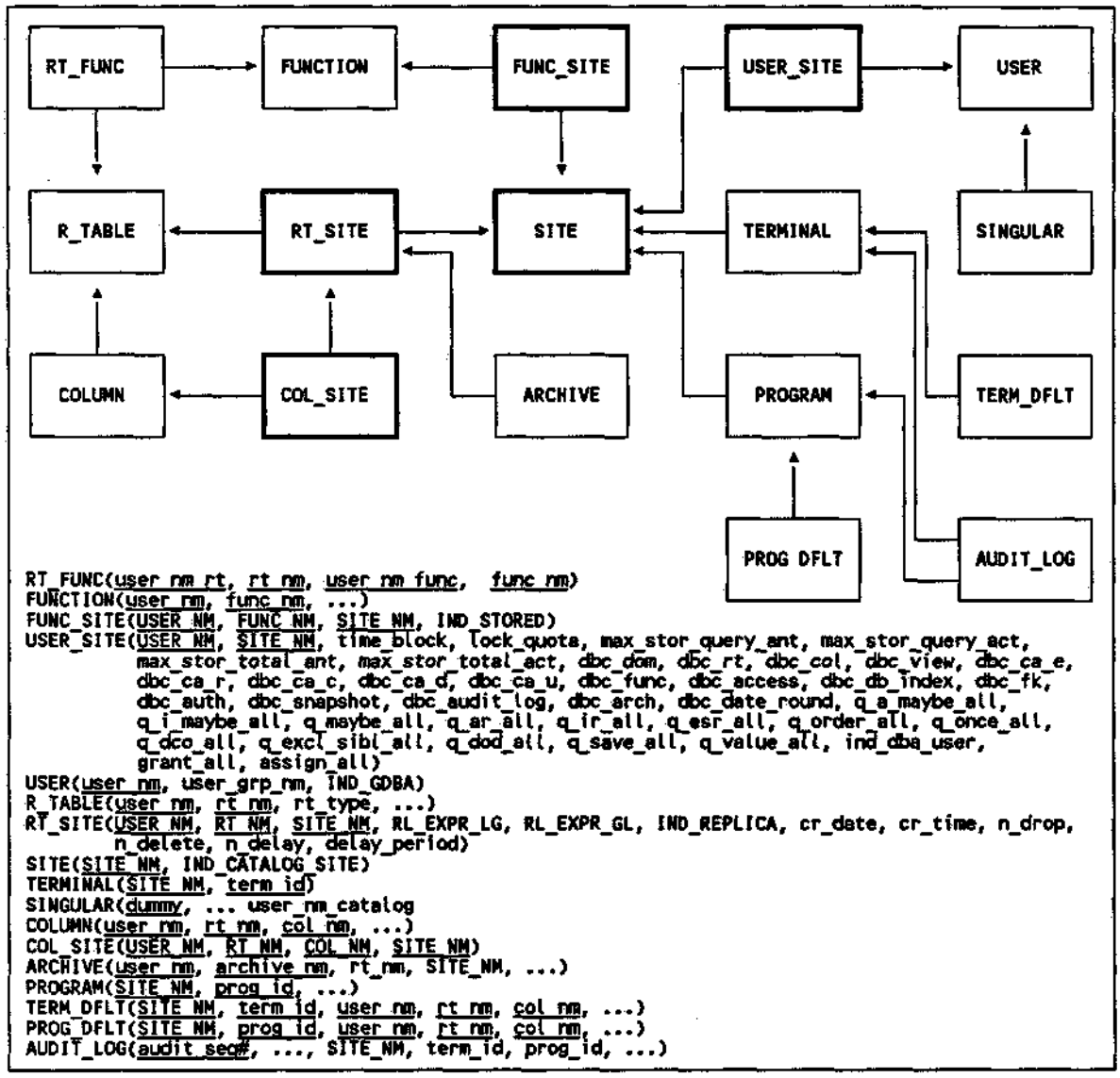


Figure 13a: Catalog support for distributed database (except conditional actions)

The core relation in figures 13a and 13c is the R-table SITE, which contains a tuple for every site that constitutes the global database. The column IND_CATALOG_SITE indicates whether the site contains a copy of the global catalog. Database objects such as R-tables, users and conditional actions are in some way related to the SITE-relation.

The catalog relation R_TABLE is not changed in any way by the introduction of database sites. Instead of adding a site identifier (SITE_NM) to the primary key, we require every R_TABLE-tuple to be referenced by at least one RT_SITE-tuple. If an R-table is known to *n* sites, there will exist *n* RT_SITE-tuples, one tuple for each site. Because base R_tables may be horizontally or vertically partitioned they will not always be assigned to one specific site. The column RL_EXPR_LG contains the local-to-global translation of the local R-table to the global database. The global R-table will be the union of the output relations of the RL_EXPR_LG-expressions applied to the various local R-tables. The RDBMS should reject any occurrence of a situation in which tuples occur with similar values for their primary key columns and differing values for their

other columns. The column RL_EXPR_LG contains an I-mark if the R-table does not physically exist at the pertinent site. The column RL_EXPR_GL contains the global-to-local translation of the global R-table to the local database user. Both expressions may only contain restrictions. Vertical partitioning (projection) is expressed by means of the R-table COL_SITE, which contains a tuple for every column of an R-table known to a given site.

For performance reasons, R-tables may be replicated at different sites. If the column IND_REPLICA in RT_SITE equals 'YES' all tuples accessible to the site's users will be physically present at the site.

> *Aside:* The presented scheme permits a relation to be both an 'original' base relation and a copy of one or more other relations. It is not clear whether Codd prescribes such flexibility. The scheme definitely departs from Codd's prescription to apply the so called 'birth site concept' in which the site identifier is a part of the R-table's identification.

Like R-tables, database users are also treated as site-independent entities. This makes it possible to introduce the same user to several sites. Unfortunately, the numerous authorization features introduced in section 11 are obviously dependent on both user and site. Therefore, the relation USER_SITE has been introduced and all non-key attributes have been transferred to this new relation. A similar change has occurred with respect to the ARCHIVE-relation (see section 4), that now references the relation RT_SITE instead of the relation R_TABLE. Another consequence of the introduction of the relation RT_SITE is the transfer of the R_TABLE-columns CR_DATE and CR_TIME to that relation (see section 4). Figure 13b rephrases the constraints introduced in sections 4 and 11 that are trivially affected by these changes. The modifications are printed in italic format.

AR02-	The combination of CR_DATE and CR_TIME in *RT_SITE* does not exceed the current system date and time.
AU01-	The values of the columns N_DROP, N_DELETE and N_DELAY in *RT_SITE* must be less than the number of individual users in the relation *USER_SITE*.
AU12-	For no *USER_SITE*-tuple the storage authorizations per query (MAX_STOR_QUERY_ANT, MAX_STOR_QUERY_ACT) exceed their total counterparts (MAX_STOR_TOTAL_ANT, MAX_STOR_TOTAL_ACT).
AU15-	Every *USER_SITE*-tuple for which IND_DBA_USER = 'YES' (*local* DBA) has the value 'YES' for GRANT_ALL.

Figure 13b: Constraints modified due to the introduction of sites

The column IND_GDBA has been added to the USER-relation to indicate whether the user is a global database administrator.

Because domains are the glue that constitutes the database schema, these objects are totally site-independent. Their definitions can be found at any site. User-defined functions are also site-independent.

Finally, figure 13a expresses the reasonable assumption that application programs and computer terminals are site-dependent. Therefore, a column SITE_NM is added to the primary key of the relations PROGRAM and TERMINAL and to the relations referencing them.

Figure 13c depicts the way in which conditional actions are supported by the catalog. Obviously it looks much like figure 7a in section 7.

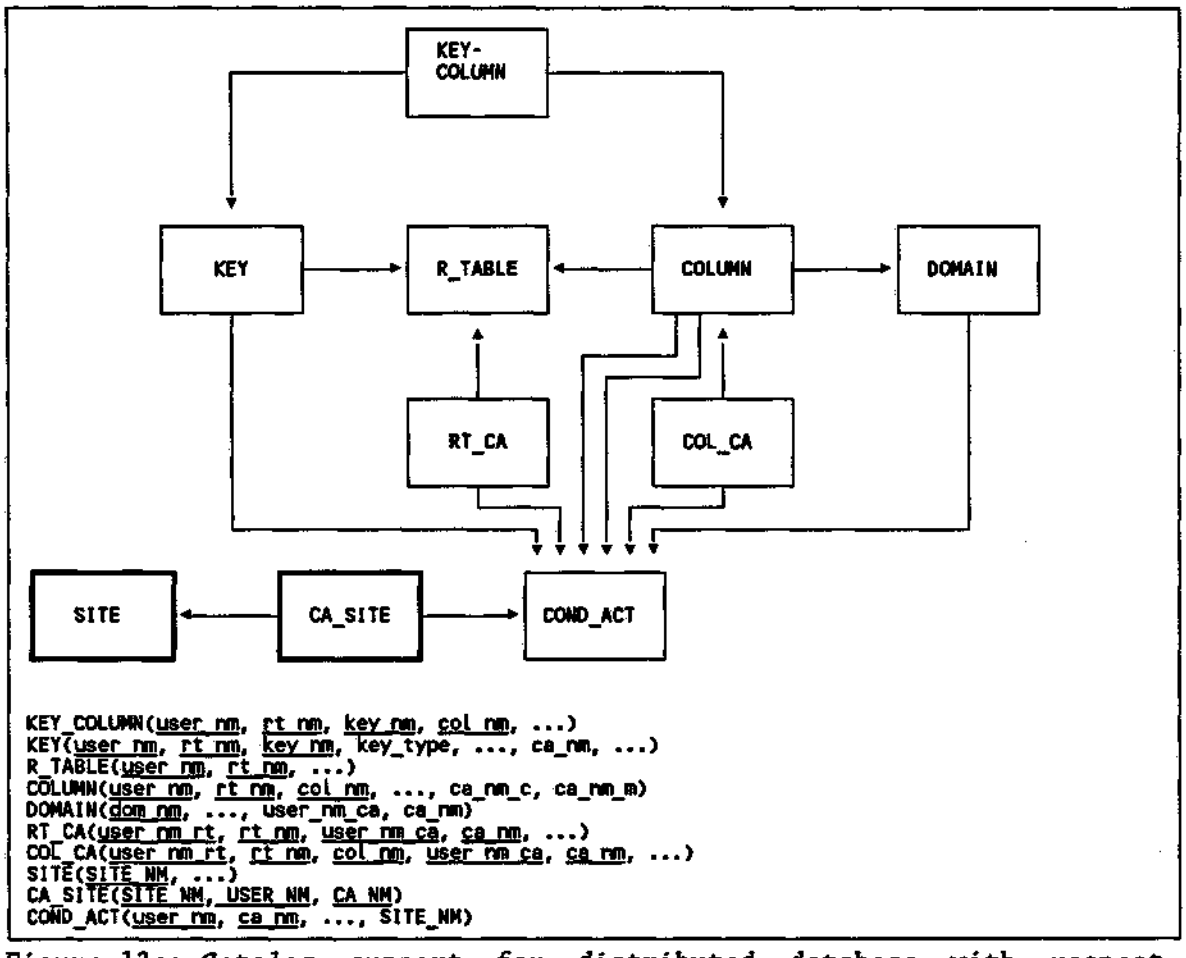


Figure 13c: Catalog support for distributed database with respect to conditional actions

Conditional actions are treated much like user-defined functions. The column SITE_NM refers to the site where the conditional action is stored. Every CA_SITE-tuple asserts the applicability of a conditional action to a site.

Figure 13d (overleaf) lists the constraints with respect to distributed database support.

DD01- Every tuple in R_TABLE is referenced by at least one RT_SITE-tuple.
DD02- Every tuple in RT_SITE is referenced by at least one COL_SITE-tuple.
DD03- The RT_SITE-columns RL_EXPR_LG, RL_EXPR_GL and IND_REPLICA must not contain I-marks if and only if the column RT_TYPE in R_TABLE = 'BASE'.
DD04- There exists at least one SITE-tuple for which IND_CATALOG_SITE equals 'YES'.
DD05- Every tuple in USER is referenced by at least one USER_SITE-tuple.
DD06- Every tuple in SITE is referenced by at least one USER_SITE-tuple.
DD07- For every individual user for which IND_GDBA = 'YES' the number of USER_SITE-tuples referencing it equals the number of SITE-tuples.
DD08- The USER-tuple referenced by the SINGULAR-tuple has the value 'YES' for IND_GDBA.
DD09- Every USER_SITE-tuple that references a USER-tuple for which IND_GDBA = 'YES' has the value 'YES' for IND_DBA_USER.
DD10- Every tuple in FUNCTION is referenced by at least one FUNC_SITE-tuple for which IND_STORED = 'YES'.
DD11- If a user-defined function operates on specific R-tables, these R-tables must be known to the site (via RT_FUNC) to which the function is assigned (via RT_SITE).
DD12- Every tuple in COND_ACT is referenced by at least one CA_SITE-tuple.
DD13- For every CA_SITE-tuple that references a COND_ACT-tuple that is referenced by a KEY-tuple, the columns that constitute the key (in KEY_COLUMN) must also be known to the site (in COL_SITE).
DD14- For every CA_SITE-tuple that references a COND_ACT-tuple that is referenced by a RT_CA-tuple, the R-tables designated by these tuples must also be known to the site (in RT_SITE).
DD15- For every CA_SITE-tuple that references a COND_ACT-tuple that is referenced by a COL_CA-tuple, the columns designated by these tuples must also be known to the site (in COL_SITE).
DD16- For every CA_SITE-tuple that references a COND_ACT-tuple that is referenced by a COLUMN-tuple, the columns designated by these tuples must also be known to the site (in COL_SITE).
DD17- For every COND_ACT-tuple that is referenced by a DOMAIN-tuple, the number of CA_SITE-tuples referencing it equals the number of SITE-tuples.

Figure 13d: Constraints for distributed database

14. DATABASE STATISTICS

The catalog should contain statistical information about the database. This information is used by the optimizer in compiling and recompiling RL commands [p.282]. Minimum statistics include at least the number of rows in each base R-table and the number of distinct values in every column. The statistics are periodically updated by the RDBMS and/or the DBA. Both can use the triggered action mechanism, as described in section 7, for this purpose.

Support for database statistics does not require significant extentions of the catalog. Figure 14a (overleaf) shows the additions needed.

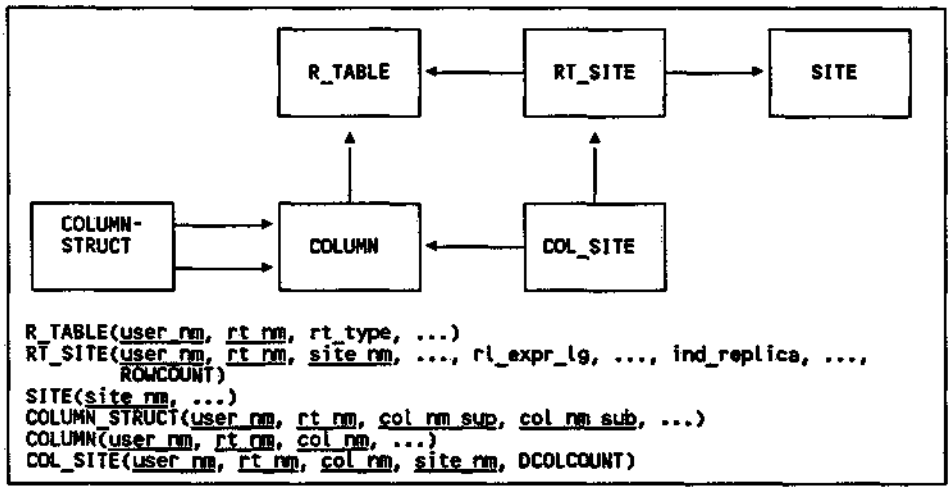


Figure 14a: Catalog support for database statistics

Column ROWCOUNT contains the number of rows of every base R-table. Column DCOLCOUNT contains the distinct number of rows for every column of such R-tables. Both counts are site-dependent. Figure 14b lists the constraints with respect to database statistics.

ST01-	The column ROWCOUNT in RT_SITE must not be I-marked if and only if IND_REPLICA = 'YES' or RL_EXPR_LG is not I-marked.
ST02-	The column DCOLCOUNT in COL_SITE must be I-marked if and only if the column ROWCOUNT in the referenced RT_SITE-tuple is also I-marked.
ST03-	The column DCOLCOUNT in COL_SITE must not exceed the column ROWCOUNT in the referenced RT_SITE-tuple.

Figure 14b: Constraints for database statistics

15. AN OVERVIEW OF THE CATALOG MODEL

For reasons of comprehensibility, the catalog model has been presented in many small pieces. Figure 15a sums up all catalog R-table descriptions in alphabetical order. Figure 15b contains references to the sections in which the R-tables play a part and indicates whether an R-table is introduced (I), changed (C) or merely showed for purposes of constraint specification (S) in the section. The '*' mark indicates that the changes are wholly or partly undone in section 13.

```
 1 ARCHIVE(user_nm, archive_nm, rt_nm, cr_date, cr_time, site_nm)
 2 ARGUMENT(user_nm, func_nm, arg_seq#, arg_nm)
 3 AUDIT_INTERVAL(user_nm, rt_nm, date_log_on, time_log_on, date_log_off, time_log_off,
                  user_nm_log_on, user_nm_log_off)
 4 AUDIT_LOG(audit_seq#, date_log, time_log, user_nm_rt, rt_nm, col_nm, user_nm_mut, site_nm,
             term_id, prog_id, db_val_old, db_val_new)
 5 BASIC_DATATYPE(bd_type_nm, ind_length, ind_scale)
 6 CA_SITE(site_nm, user_nm, ca_nm)
 7 COLUMN(user_nm, rt_nm, col_nm, col_seq#, dom_nm, a_mark_all, i_mark_all, upd_all, user_nm_base,
          rt_nm_base, col_nm_base, bd_type_nm, length, scale, ca_nm_c, ca_nm_m, mark_pref)
 8 COLUMN_STRUCT(user_nm, rt_nm, col_nm_sup, col_nm_sub, cs_seq#)
 9 COL_AUTH(user_nm_rt, rt_nm, col_nm, user_nm_grantor, user_nm_grantee, upd_all, rl_expr_upd,
            default_val, i_mark_all, i_unmark_all,
10 COL_CA(user_nm_rt, rt_nm, col_nm, user_nm_ca, ca_nm, group_id)
11 COL_SITE(user_nm, rt_nm, col_nm, site_nm, dcolcount)
12 COND_ACT(user_nm, ca_nm, condition, ca_date, ca_time, interval, ttime, ind_static, ta_exec,
            constr_type, site_nm)
13 DB_INDEX(user_nm, index_nm, dom_nm)
14 DB_INDEX_COL(user_nm_dbi, index_nm, user_nm_rt, rt_nm, col_nm)
15 DOMAIN(dom_nm, dom_type, comp_all, bd_type_nm, length, scale, user_nm_dom, user_nm_ca, ca_nm)
16 DOMAIN_STRUCT(dom_nm_sup, dom_nm_sub, ds_seq#)
17 DOM_RANGE(dom_nm, dr_seq#, loval, hival)
18 FUNCTION(user_nm, func_nm, host_lang, rout_src, rout_exec, ind_dod, func_nm_inv, func_nm_dod)
19 FUNC_SITE(user_nm, func_nm, site_nm, ind_stored)
20 KEY(user_nm, rt_nm, key_nm, key_type, index_nm, ca_nm, upd_act, del_act)
21 KEY_COLUMN(user_nm, rt_nm, key_nm, col_nm, key_col_seq#)
22 KEY_REF(user_nm_pk, rt_nm_pk, user_nm_fk, rt_nm_fk, key_nm)
23 LOG_PK_COLUMN(audit_seq#, user_nm, rt_nm, col_nm, pk_val)
24 PROGRAM(site_nm, prog_id, time_block, lock_quota)
25 PROG_DFLT(site_nm, prog_id, user_nm, rt_nm, col_nm, default_val)
26 RL_KEYWORD(rl_keyw)
27 RT_AUTH(user_nm_rt, rt_nm, user_nm_grantor, user_nm_grantee, ret_all, ins_all, pk_upd_all,
         arch_all, del_all, snapshot_all, checkout_all, pk_hidden, rl_expr_ret, rl_expr_ins,
         rl_expr_pk_upd, rl_expr_arch, rl_expr_del, rl_expr_snapshot, rl_expr_checkout)
28 RT_CA(user_nm_rt, rt_nm, user_nm_ca, ca_nm, ins, del, ret)
29 RT_FUNC(user_nm_rt, rt_nm, user_nm_func, func_nm)
30 RT_SITE(user_nm, rt_nm, site_nm, rl_expr_lg, rl_expr_gl, ind_replica, cr_date, cr_time, n_drop,
          n_delete, n_delay, delay_period, rowcount)
31 RT_SYN(user_nm_syn, rt_nm_syn, user_nm_rt, rt_nm)
32 R_TABLE(user_nm, rt_nm, rt_type, rl_expr, ins_all, del_all, func_nm_vw)
33 SINGULAR(dummy, dom_nm_func, dom_nm_arg, user_nm_catalog)
34 SITE(site_nm, ind_catalog_site)
35 TERMINAL(site_nm, term_id)
36 TERM_DFLT(site_nm, term_id, user_nm, rt_nm, col_nm, default_val)
37 USER(user_nm, user_grp_nm, ind_gdba)
38 USER_SITE(user_nm, site_nm, time_block, lock_quota, max_stor_query_ant, max_stor_query_act,
       max_stor_total_ant, max_stor_total_act, dbc_dom, dbc_rt, dbc_col, dbc_view, dbc_ca_e,
       dbc_ca_r, dbc_ca_c, dbc_ca_d, dbc_ca_u, dbc_func, dbc_access, dbc_db_index, dbc_fk,
       dbc_auth, dbc_snapshot, dbc_audit_log, dbc_arch, dbc_date_round, q_a_maybe_all,
       q_i_maybe_all, q_maybe_all, q_ar_all, q_ir_all, q_esr_all, q_order_all, q_once_all,
       q_dco_all, q_excl_sibl_all, q_dod_all, q_save_all, q_value_all, ind_dba_user, grant_all,
       assign_all)
```

Figure 15a: Descriptions of all catalog R-tables

	2	3	4	5	6	7	8	9	10	11	12	13	14
1 ARCHIVE			I						C			C	
2 ARGUMENT							I		C				
3 AUDIT_INTERVAL											I		
4 AUDIT_LOG											I	C	
5 BASIC_DATATYPE								I					
6 CA_SITE												I	
7 COLUMN	I	S		C	S	C		S	C	C	S	S	S
8 COLUMN_STRUCT		I						S	C				S
9 COL_AUTH										I			
10 COL_CA						I			C			S	
11 COL_SITE												I	C
12 COND_ACT						I			C			C	
13 DB_INDEX					I				C				
14 DB_INDEX_COL					I				C				
15 DOMAIN	I	S		S	S	C	S	S	C	S		S	
16 DOMAIN_STRUCT		I											
17 DOM_RANGE	I						S			S			
18 FUNCTION							I	S	C			C	
19 FUNC_SITE												I	
20 KEY	I		S		C	C			C			S	
21 KEY_COLUMN	I				S				C			S	
22 KEY_REF	I								C				
23 LOG_PK_COLUMN											I		
24 PROGRAM										I	S	C	
25 PROG_DFLT										I		C	
26 RL_KEYWORD								I					
27 RT_AUTH										I			
28 RT_CA						I			C			S	
29 RT_FUNC							I		C			S	
30 RT_SITE												I	C
31 RT_SYN								I	C				
32 R_TABLE	I		C*	C	S	S	C	C	C	C*	S	C	S
33 SINGULAR							I		C			S	
34 SITE												I	S
35 TERMINAL										I	S	C	
36 TERM_DFLT										I		C	
37 USER									I	C*	S	C	
38 USER_SITE												I	

Figure 15b: Use of catalog R-tables in the previous sections.

16. CONCLUDING REMARKS

Considering the vast scope and complexity of Codd's RM/V2-proposal, it is unlikely that this manifesto can serve as much more than a first approximation of a catalog standard for future RM/V2 DBMS's. If it is acceptable as such, the main objective of the manifesto is attained. Nevertheless, the authors hope that the highly technical nature of the manifesto does not obstruct our objective to convince the reader that a catalog standard is extremely desirable for a large number of reasons. Moreover, the authors hope that the unavoidable lengthy and technical discussion of the catalog standard aspects does not obscure the fact that basically the Relational Model is a very simple model. Its core features can be described in the fairly simple schema introduced in section 2. Comparing RM/V2 as a whole to other data models is generally unfair, because most of these data models do not match RM/V2's scope and level of detail.

An interesting question is whether RM/V2 will be accepted in the market-place. It is conceivable that today's RDBMSs will be succeeded by products based on another version of the Relational Model or on a non-relational

data model. No doubt researchers and practitioners will criticize aspects of RM/V2 as the authors do in appendix C. It is our feeling that, provided the future of database management is relational, this manifesto can still be useful in discussions about the form of a relational catalog standard. The effects of proposed deviations from Codd's RM/V2 proposal on the catalog's structure and complexity may even be used as one more basis for such discussions.

This last observation leads to the conclusion that the form in which the catalog standard is expressed is more important than it may seem. If one looks upon the model as *just another relational database*, one drops the artificial and arbitrary distinction between data and metadata and invites database designers to design their databases as an extension to the catalog model, thereby creating the opportunity to design applications that are highly reusable and flexible. Another consequence is that the catalog standard becomes a database design standard at the same time, because it implicitly shows what a complete relational database design looks like. Viewing the catalog as just a database may also serve to convince RDBMS vendors that Data Definition Language (DDL) and Data Control Language (DCL) are entirely redundant, because DDL- and DCL-statements can be translated to one or more DML-statements on the catalog database [BUIT91] [VELD91a] [WARD90].

Perhaps unnecessarily, we must stress that the catalog standard is not a proposal for the way in which an RM/V2-DBMS catalog should be structured internally. A catalog standard only prescribes the way in which catalog data are presented to a terminal user or an application program. Nevertheless, if the vendor of an RM/V2-DBMS makes use of performance-enhancing features, that are not a part of RM/V2, it would be at least elegant to offer these features to the RDBMS's users too. As a result, non-RM/V2 features will be part of the RDBMS's catalog. As long as the features apply to lower levels than the relational level there can be no objection to that. To conclude, the catalog standard is a minimum standard as far as subrelational aspects of RM/V2 are concerned.

Finally, although the authors have tried to be thorough, there is probably ample room for improvements of the model. Improvements may take place with respect to naming of database objects, omissions, and errors of interpretation. Furthermore, there are instances in which either Codd is not clear or the Relational Model provides multiple design options, non of which is clearly superior to the others. In these respects this manifesto is an invitation to dr. Codd to clarify RM/V2 and to keep working on the third version of the Relational Model.

References:

[BOOG91] **Boogaard M, Veldwijk R J, Spoor E.R.K. and Dijk M V van,** 'On Generalization in the Relational Model' *Research memorandum 1991-37 Vrije Universiteit, Amsterdam* (1991)

[BUIT91] **Buitendijk R B and Lek H van der,** 'Direct Manipulation of a Data Dictionary with SQL' *Information Systems* Vol 16 No 3 (1991) pp. 323-333

[CODD70] **Codd E F** 'A Relational Model of Data for Large Shared Data Banks' *Communications of the ACM* Vol 13 No 6 (1970)

[CODD81] **Codd E F** 'Data Models in Database Management' *ACM SIGMOD* Record 11 No 2 (february 1981)

[CODD90] **Codd E F** *The Relational Model for Database Management, Version 2* Reading, Massachusetts: Addison-Wesley Publishing Company (1990)

[CURT81] **Curtice R M** 'DATA DICTIONARIES: an assessment of current practice and problems' *Proceedings of the 7th International Conference on Very Large Databases* pp. 564-570 (1981)

[DATE83] **Date C J** 'A Formal Definition of the Relational Model' in *Relational database, Selected Writings* Reading, Massachusetts: Addison-Wesley Publishing Company (1989)

[MCGE76] **McGee W C** 'On User Criteria for Data Model Evaluation' *ACM Transactions on Database Systems* Vol 1 No 4 (1976)

[NIJS89] **Nijssen G M and Halpin T A** *Conceptual Schema and Relational Database Design, A Fact Oriented Approach* Sydney, Australia: Prentice Hall (1989)

[ROSS81] **Ross R G** *Data Dictionaries and Data Administration* AMACOM (1981)

[SHNE82] **Shneiderman B and Thomas G** 'An Architecture for Automatic Relational Database System Conversion' *ACM Transaction on Database Systems* Vol 7 No 2 (1982)

[SNOD87] **Snodgrass R** 'The Temporal Query Language TQuel' *ACM Transactions on Database Systems* Vol 12 pp. 247-298 (1987)

[TSIC82] **Tsichritzis D C and Lochovsky F H** *Data Models* New Jersey, Prentice Hall (1982)

[VELD90] **Veldwijk R J, Dijk M V van, Boogaard M and Spoor E R K** 'Generalisatie in een Relationele Omgeving: Een benadering gericht op het terugdringen van onderhoud' *Research memorandum 1990-57 Vrije Universiteit, Amsterdam* (1990)

[VELD91a] **Veldwijk R J, Boogaard M, Dijk M V van and Spoor E R K** 'EDSOs, Implosion and Explosion: Concepts to Automate a Part of Application Maintenance in Relational Databases' *Information and Software Technology* Vol 33 No 5 (june 1991)

[VELD91b] **Veldwijk R J, Spoor E R K, Dijk M V van and Boogaard M** 'On The Expressive Power of the Relational Model, a Database Designers Point of View' *Research memorandum 1991-59 Vrije Universiteit, Amsterdam* (1991). To appear in the proceedings of the 10th International Conference on Information systems New York December 1991

[WARD90] **Warden A** 'Chivalry', In C.J. Date *Relational Database, Writings 1985 -1989* Reading, Massachusetts: Addison-Wesley Publishing Company, 1990.

Appendix A Contents of the Catalog R-tables

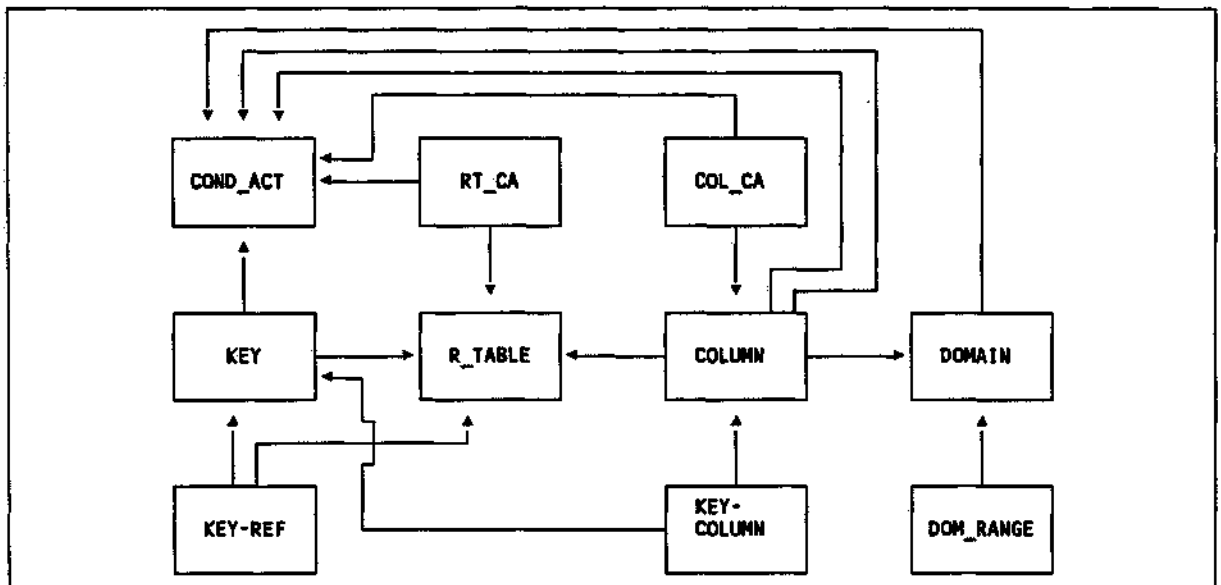
This appendix defines the catalog model in terms of its own database definition. Although such a definition is highly technical and hard to interpret, it is entirely formal and unambiguous. Moreover, the catalog self-description contains information about the catalog that has not been discussed in the main text. The manifesto concentrates on the design decisions with respect to the catalog's structure and thus contains *know-why* information. This appendix and appendix B contain an implementation of the RM/V2-catalog and thus provide *know-how* information. The authors have implemented the catalog model using the ORACLE™ DBMS.

An implementation of the catalog with its own design does not require all R-tables described in the catalog to be filled. For instance, the R-table COLUMN_STRUCT (see section 3) is not relevant here because composite columns were not used in the description of the catalog standard. The R-tables we refer to are the relations ARCHIVE, AUDIT_INTERVAL, AUDIT_LOG, COLUMN_STRUCT, COL_AUTH, DB_INDEX, DB_INDEX_COL, DOMAIN_STRUCT, LOG_PK_COLUMN, PROGRAM, PROG_DFLT, RT_AUTH, RT_FUNC, RT_SYN, TERMINAL, and TERM_DFLT.

Other R-tables must be filled to realize a working catalog, but these R-tables do not provide the technically interested reader with a deeper insight in the catalog model. For instance, the R-table BASIC_DATATYPE must be filled with tuples describing the datatypes known to the RDBMS, but these descriptions are not relevant to the manifesto. The R-tables we refer to are the relations ARGUMENT, BASIC_DATATYPE, CA_SITE, COL_SITE, FUNCTION, FUNC_SITE, RL_KEYWORD, RT_SITE, SINGULAR, SITE, USER and USER_SITE.

The contents of the remaining R-tables are relevant to the purpose of this appendix. These R-tables were introduced and discussed in sections 2 and 7. Figure 16 (overleaf), which is a combination of figures 2a and 7a, redisplays the relevant data structures.

Not all columns in these R-tables are relevant for this appendix. For instance, the columns USER_NM are not interesting because all database objects belong to one single user, namely the user that owns the catalog. In this appendix we shall only show the contents of those columns that contain information that cannot be found in the manifesto. In figure 16, these columns are shown in upper case.



COND_ACT(user_nm, CA_NM, condition, ca_date, ca_time, interval, TTIME, ind_static, ta_exec,
CONSTR_TYPE, site_nm)
RT_CA(user_nm_rt, RT_NM, user_nm_ca, CA_NM, INS, DEL, ret)
COL_CA(user_nm_rt, RT_NM, COL_NM, user_nm_ca, CA_NM, GROUP_ID)
KEY(user_nm, RT_NM, KEY_NM, KEY_TYPE, index_nm, CA_NM, UPD_ACT, DEL_ACT)
R_TABLE(user_nm, RT_NM, RT_TYPE, rl_expr, ins_all, del_all, func_nm_vw)
COLUMN(user_nm, RT_NM, COL_NM, COL_SEQ#, DOM_NM, A_MARK_ALL, I_MARK_ALL, upd_all, user_nm_base,
rt_nm_base, col_nm_base, bd_type_nm, length, scale, CA_NM_C, CA_NM_M, MARK_PREF)
DOMAIN(DOM_NM, DOM_TYPE, COMP_ALL, BD_TYPE_NM, LENGTH, SCALE, user_nm_dom, user_nm_ca, CA_NM)
KEY_REF(user_nm_pk, RT_NM_PK, user_nm_fk, RT_NM_FK, KEY_NM)
KEY_COLUMN(user_nm, RT_NM, KEY_NM, COL_NM, KEY_COL_SEQ#)
DOM_RANGE(DOM_NM, DR_SEQ#, LOVAL, HIVAL)

Figure 16: R-tables relevant to appendix A

Although the column CONDITION in COND_ACTION is relevant to this appendix, it is depicted in lower case because the conditions of the the constraints are listed in appendix B. Because all conditional actions are static database constraints, columns CA_DATE, CA_TIME, INTERVAL and IND_STATIC in COND_ACTION and RET in RT_CA are irrelevant. Because all catalog R-tables are base R-tables, columns RL_EXPR, INS_ALL, DEL_ALL and FUNC_NM_VW are irrelevant. For the same reason many columns in COLUMN are irrelevant.

The relation R_TABLE is the least interesting of the tables depicted hereafter because all catalog R-tables are simple base R-tables. For reasons of completeness it is nevertheless included in the appendix.

The R_TABLE data are retrieved by means of the following SQL statement:

```
SELECT RT_NM, RT_TYPE
FROM R_TABLE
WHERE USER_NM = 'RMV2'    /* username of catalog owner */
ORDER BY RT_NM
```

R_TABLE

RT_NM	RT_TYPE
ARCHIVE	BASE
ARGUMENT	BASE
AUDIT_INTERVAL	BASE
AUDIT_LOG	BASE
BASIC_DATATYPE	BASE
CA_SITE	BASE
COLUMN	BASE
COLUMN_STRUCT	BASE
COL_AUTH	BASE
COL_CA	BASE
COL_SITE	BASE
COND_ACT	BASE
DB_INDEX	BASE
DB_INDEX_COL	BASE
DOMAIN	BASE
DOMAIN_STRUCT	BASE
DOM_RANGE	BASE
FUNCTION	BASE
FUNC_SITE	BASE

RT_NM	RT_TYPE
KEY	BASE
KEY_COLUMN	BASE
KEY_REF	BASE
LOG_PK_COLUMN	BASE
PROGRAM	BASE
PROG_DFLT	BASE
RL_KEYWORD	BASE
RT_AUTH	BASE
RT_CA	BASE
RT_FUNC	BASE
RT_SITE	BASE
RT_SYN	BASE
R_TABLE	BASE
SINGULAR	BASE
SITE	BASE
TERMINAL	BASE
TERM_DFLT	BASE
USER	BASE
USER_SITE	BASE

Aside: Because the R-table names COLUMN and USER are reserved words in an ORACLE-environment these R-tables have been implemented under different names.

The COLUMN data are retrieved by means of the following SQL statement:

```
SELECT RT_NM,
       COL_SEQ# C#,
       COL_NM,
       DOM_NM,
       A_MARK_ALL A,
       I_MARK_ALL I,
       CA_NM_C,
       CA_NM_M,
       MARK_PREF P
FROM COLUMN
WHERE USER_NM = 'RMV2'   /* username of catalog owner */
ORDER BY RT_NM, COL_SEQ#
```

COLUMN

RT_NM	C#	COL_NM	DOM_NM	A	I	CA_NM_C	CA_NM_M	P
ARCHIVE	1	USER_NM	USER_NM	NO	NO			
	2	ARCHIVE_NM	ARCHIVE_NM	NO	NO			
	3	RT_NM	RT_NM	NO	NO		M_ARC_RT_NM	
	4	CR_DATE	DATE	NO	NO		M_ARC_CR_DATE	
	5	CR_TIME	TIME	NO	NO		M_ARC_CR_TIME	
	6	SITE_NM	SITE_NM	NO	NO		M_ARC_SITE_NM	
ARGUMENT	1	USER_NM	USER_NM	NO	NO			
	2	FUNC_NM	FUNC_NM	NO	NO			
	3	ARG_SEQ#	ARG_SEQ#	NO	NO			
	4	ARG_NM	ARG_NM	NO	NO		M_ARG_NM	
AUDIT_INTERVAL	1	USER_NM	USER_NM	NO	NO			
	2	RT_NM	RT_NM	NO	NO			
	3	DATE_LOG_ON	DATE	NO	NO			
	4	TIME_LOG_ON	TIME	NO	NO			
	5	DATE_LOG_OFF	DATE	YES	NO		M_DATE_LOG_OFF	
	6	TIME_LOG_OFF	TIME	YES	NO		M_TIME_LOG_OFF	
	7	USER_NM_LOG_ON	USER_NM	NO	NO		M_USER_NM_LOG_ON	
	8	USER_NM_LOG_OFF	USER_NM	YES	NO		M_USER_NM_LOG_OFF	

COLUMN (continued)

RT_NM	C#	COL_NM	DOM_NM	A	I	CA_NM_C	CA_NM_M	P
AUDIT_LOG	1	AUDIT_SEQ#	AUDIT_SEQ#	NO	NO			
	2	DATE_LOG	DATE	NO	NO		M_DATE_LOG	
	3	TIME_LOG	TIME	NO	NO		M_TIME_LOG	
	4	USER_NM_RT	USER_NM	NO	NO		M_AL_USER_NM_RT	
	5	RT_NM	RT_NM	NO	NO		M_AL_RT_NM	
	6	COL_NM	COL_NM	NO	NO		M_AL_COL_NM	
	7	USER_NM_MUT	USER_NM	YES	NO		M_USER_NM_MUT	
	8	SITE_NM	SITE_NM	YES	NO		M_AL_SITE_NM	
	9	TERM_ID	TERM_ID	YES	NO		M_AL_TERM_ID	
	10	PROG_ID	PROG_ID	YES	NO		M_AL_PROG_ID	
	11	DB_VAL_OLD	DB_VAL	YES	NO		M_DB_VAL_OLD	
	12	DB_VAL_NEW	DB_VAL	YES	NO		M_DB_VAL_NEW	
BASIC_DATATYPE	1	BD_TYPE_NM	BD_TYPE_NM	NO	NO			
	2	IND_LENGTH	IND_LENGTH	NO	NO		M_IND_LENGTH	
	3	IND_SCALE	IND_SCALE	NO	NO		M_IND_SCALE	
CA_SITE	1	USER_NM	USER_NM	NO	NO			
	2	CA_NM	CA_NM	NO	NO			
	3	SITE_NM	SITE_NM	NO	NO			
COLUMN	1	USER_NM	USER_NM	NO	NO			
	2	RT_NM	RT_NM	NO	NO			
	3	COL_NM	COL_NM	NO	NO			
	4	COL_SEQ#	COL_SEQ#	NO	NO		M_COL_SEQ#	
	5	DOM_NM	DOM_NM	YES	NO		M_COL_DOM_NM	
	6	A_MARK_ALL	IND_MARK	NO	YES		M_A_MARK_ALL	
	7	I_MARK_ALL	IND_MARK	NO	YES		M_COL_I_MARK_ALL	
	8	UPD_ALL	UPD_ALL	NO	YES		M_COL_UPD_ALL	
	9	USER_NM_BASE	USER_NM	YES	NO		M_USER_NM_BASE	
	10	RT_NM_BASE	RT_NM	YES	NO		M_RT_NM_BASE	
	11	COL_NM_BASE	COL_NM	YES	NO		M_COL_NM_BASE	
	12	BD_TYPE_NM	BD_TYPE_NM	YES	NO		M_COL_BD_TYPE_NM	
	13	LENGTH	LENGTH	NO	YES		M_COL_LENGTH	
	14	SCALE	SCALE	NO	YES		M_COL_SCALE	
	15	CA_NM_C	CA_NM	YES	NO		M_CA_NM_C	
	16	CA_NM_M	CA_NM	YES	NO		M_CA_NM_M	
	17	MARK_PREF	MARK_PREF	NO	YES		M_MARK_PREF	
COLUMN_STRUCT	1	USER_NM	USER_NM	NO	NO			
	2	RT_NM	RT_NM	NO	NO			
	3	COL_NM_SUP	COL_NM	NO	NO			
	4	COL_NM_SUB	COL_NM	NO	NO			
	5	CS_SEQ#	CS_SEQ#	NO	NO		M_CS_SEQ#	
COL_AUTH	1	USER_NM_RT	USER_NM	NO	NO			
	2	RT_NM	RT_NM	NO	NO			
	3	COL_NM	COL_NM	NO	NO			
	4	USER_NM_GRANTOR	USER_NM	NO	NO			
	5	USER_NM_GRANTEE	USER_NM	NO	NO			
	6	UPD_ALL	UPD_ALL	NO	YES		M_CAU_UPD_ALL	
	7	RL_EXPR_UPD	RL_EXPR	NO	YES	C_RL_EXPR_UPD	M_RL_EXPR_UPD	
	8	DEFAULT_VAL	DB_VAL	YES	YES			I
	9	I_MARK_ALL	IND_MARK	NO	YES		M_CAU_I_MARK_ALL	
	10	I_UNMARK_ALL	IND_UNMARK	NO	YES		M_I_UNMARK_ALL	
COL_CA	1	USER_NM_RT	USER_NM	NO	NO			
	2	RT_NM	RT_NM	NO	NO			
	3	COL_NM	COL_NM	NO	NO			
	4	USER_NM_CA	USER_NM	NO	NO			
	5	CA_NM	CA_NM	NO	NO			
	6	GROUP_ID	GROUP_ID	NO	YES		M_GROUP_ID	
COL_SITE	1	USER_NM	USER_NM	NO	NO			
	2	RT_NM	RT_NM	NO	NO			
	3	COL_NM	COL_NM	NO	NO			
	4	SITE_NM	SITE_NM	NO	NO			
	5	DCOLCOUNT	ROW_COL_COUNT	NO	YES		M_DCOLCOUNT	
COND_ACT	1	USER_NM	USER_NM	NO	NO			
	2	CA_NM	CA_NM	NO	NO			
	3	CONDITION	RL_EXPR	NO	YES	C_CONDITION	M_CONDITION	
	4	CA_DATE	CA_DATE	NO	YES		M_CA_DATE	
	5	CA_TIME	CA_TIME	NO	YES		M_CA_TIME	
	6	INTERVAL	INTERVAL	NO	YES		M_INTERVAL	
	7	TTIME	TTIME	NO	YES		M_TTIME	
	8	IND_STATIC	IND_STATIC	NO	NO		M_IND_STATIC	
	9	TA_EXEC	ROUTINE_NM	NO	YES		M_TA_EXEC	
	10	CONSTR_TYPE	CONSTR_TYPE	NO	YES		M_CONSTR_TYPE	
	11	SITE_NM	SITE_NM	NO	NO		M_CA_SITE_NM	
DB_INDEX	1	USER_NM	USER_NM	NO	NO			
	2	INDEX_NM	INDEX_NM	NO	NO			
	3	DOM_NM	DOM_NM	NO	NO		M_DBI_DOM_NM	

COLUMN (continued)

RT_NM	C#	COL_NM	DOM_NM	A	I	CA_NM_C	CA_NM_M	P
DB_INDEX_COL	1	USER_NM_DBI	USER_NM	NO	NO			
	2	INDEX_NM	INDEX_NM	NO	NO			
	3	USER_NM_RT	USER_NM	NO	NO			
	4	RT_NM	RT_NM	NO	NO			
	5	COL_NM	COL_NM	NO	NO			
DOMAIN	1	DOM_NM	DOM_NM	NO	NO			
	2	DOM_TYPE	DOM_TYPE	NO	NO		M_DOM_TYPE	
	3	COMP_ALL	COMP_ALL	NO	NO		M_COMP_ALL	
	4	BD_TYPE_NM	BD_TYPE_NM	YES	NO		M_DOM_BD_TYPE_NM	
	5	LENGTH	LENGTH	NO	YES		M_DOM_LENGTH	
	6	SCALE	SCALE	NO	YES		M_DOM_SCALE	
	7	USER_NM_DOM	USER_NM	NO	NO		M_USER_NM_DOM	
	8	USER_NM_CA	USER_NM	YES	NO		M_DOM_USER_NM_CA	
	9	CA_NM	CA_NM	YES	NO		M_DOM_CA_NM	
DOMAIN_STRUCT	1	DOM_NM_SUP	DOM_NM	NO	NO			
	2	DOM_NM_SUB	DOM_NM	NO	NO			
	3	DS_SEQ#	DS_SEQ#	NO	NO		M_DS_SEQ#	
DOM_RANGE	1	DOM_NM	DOM_NM	NO	NO			
	2	DR_SEQ#	DR_SEQ#	NO	NO			
	3	LOVAL	DB_VAL	NO	NO		M_LOVAL	
	4	HIVAL	DB_VAL	NO	NO		M_HIVAL	
FUNCTION	1	USER_NM	USER_NM	NO	NO			
	2	FUNC_NM	FUNC_NM	NO	NO			
	3	HOST_LANG	HOST_LANG	NO	YES		M_HOST_LANG	
	4	ROUT_SRC	ROUTINE_NM	NO	YES		M_ROUT_SRC	
	5	ROUT_EXEC	ROUTINE_NM	NO	YES		M_ROUT_EXEC	
	6	IND_DOD	IND_DOD	NO	YES		M_IND_DOD	
	7	FUNC_NM_INV	FUNC_NM	YES	NO		M_FUNC_NM_INV	
	8	FUNC_NM_DOD	FUNC_NM	YES	NO		M_FUNC_NM_DOD	
FUNC_SITE	1	USER_NM	USER_NM	NO	NO			
	2	FUNC_NM	FUNC_NM	NO	NO			
	3	SITE_NM	SITE_NM	NO	NO			
	4	IND_STORED	IND_STORED	NO	NO		M_IND_STORED	
KEY	1	USER_NM	USER_NM	NO	NO			
	2	RT_NM	RT_NM	NO	NO			
	3	KEY_NM	KEY_NM	NO	NO			
	4	KEY_TYPE	KEY_TYPE	NO	NO		M_KEY_TYPE	
	5	INDEX_NM	INDEX_NM	YES	NO		M_KEY_INDEX_NM	
	6	CA_NM	CA_NM	YES	NO		M_KEY_CA_NM	
	7	UPD_ACT	FK_RULE	YES	NO		M_UPD_ACT	
	8	DEL_ACT	FK_RULE	YES	NO		M_DEL_ACT	
KEY_COLUMN	1	USER_NM	USER_NM	NO	NO			
	2	RT_NM	RT_NM	NO	NO			
	3	KEY_NM	KEY_NM	NO	NO			
	4	COL_NM	COL_NM	NO	NO			
	5	KEY_COL_SEQ#	KEY_COL_SEQ#	NO	NO		M_KEY_COL_SEQ#	
KEY_REF	1	USER_NM_PK	USER_NM	NO	NO			
	2	RT_NM_PK	RT_NM	NO	NO			
	3	USER_NM_FK	USER_NM	NO	NO			
	4	RT_NM_FK	RT_NM	NO	NO			
	5	KEY_NM	KEY_NM	NO	NO			
LOG_PK_COLUMN	1	AUDIT_SEQ#	AUDIT_SEQ#	NO	NO			
	2	USER_NM	USER_NM	NO	NO			
	3	RT_NM	RT_NM	NO	NO			
	4	COL_NM	COL_NM	NO	NO			
	5	PK_VAL	DB_VAL	NO	NO		M_PK_VAL	
PROGRAM	1	SITE_NM	SITE_NM	NO	NO			
	2	PROG_ID	PROG_ID	NO	NO			
	3	TIME_BLOCK	TIME_BLOCK	YES	NO		M_PRG_TIME_BLOCK	
	4	LOCK_QUOTA	LOCK_QUOTA	NO	NO		M_PRG_LOCK_QUOTA	
PROG_DFLT	1	SITE_NM	SITE_NM	NO	NO			
	2	PROG_ID	PROG_ID	NO	NO			
	3	USER_NM	USER_NM	NO	NO			
	4	RT_NM	RT_NM	NO	NO			
	5	COL_NM	COL_NM	NO	NO			
	6	DEFAULT_VAL	DB_VAL	NO	NO		M_PDF_DEFAULT_VAL	
RL_KEYWORD	1	RL_KEYW	RL_KEYW	NO	NO			

COLUMN (continued)

RT_NM	C#	COL_NM	DOM_NM	A	I	CA_NM_C	CA_NM_M	P
RT_AUTH	1	USER_NM_RT	USER_NM	NO	NO			
	2	RT_NM	RT_NM	NO	NO			
	3	USER_NM_GRANTOR	USER_NM	NO	NO			
	4	USER_NM_GRANTEE	USER_NM	NO	NO			
	5	RET_ALL	RET_ALL	NO	NO		M_RET_ALL	
	6	INS_ALL	INS_ALL	NO	YES		M_RTA_INS_ALL	
	7	PK_UPD_ALL	PK_UPD_ALL	NO	YES		M_PK_UPD_ALL	
	8	ARCH_ALL	ARCH_ALL	NO	NO		M_ARCH_ALL	
	9	DEL_ALL	DEL_ALL	NO	YES		M_RTA_DEL_ALL	
	10	SNAPSHOT_ALL	SNAPSHOT_ALL	NO	NO		M_SNAPSHOT_ALL	
	11	CHECKOUT_ALL	CHECKOUT_ALL	NO	YES		M_CHECKOUT_ALL	
	12	PK_HIDDEN	PK_HIDDEN	NO	YES		M_PK_HIDDEN	
	13	RL_EXPR_RET	RL_EXPR	NO	YES	C_RL_EXPR_RET	M_RL_EXPR_RET	
	14	RL_EXPR_INS	RL_EXPR	NO	YES	C_RL_EXPR_INS	M_RL_EXPR_INS	
	15	RL_EXPR_PK_UPD	RL_EXPR	NO	YES	C_RL_EXPR_PK_UPD	M_RL_EXPR_PK_UPD	
	16	RL_EXPR_ARCH	RL_EXPR	NO	YES	C_RL_EXPR_ARCH	M_RL_EXPR_ARCH	
	17	RL_EXPR_DEL	RL_EXPR	NO	YES	C_RL_EXPR_DEL	M_RL_EXPR_DEL	
	18	RL_EXPR_SNAPSHOT	RL_EXPR	NO	YES	C_RL_EXPR_SNAPSHOT	M_RL_EXPR_SNAPSHOT	
	19	RL_EXPR_CHECKOUT	RL_EXPR	NO	YES	C_RL_EXPR_CHECKOUT	M_RL_EXPR_CHECKOUT	
RT_CA	1	USER_NM_RT	USER_NM	NO	NO			
	2	RT_NM	RT_NM	NO	NO			
	3	USER_NM_CA	USER_NM	NO	NO			
	4	CA_NM	CA_NM	NO	NO			
	5	INS	INS_ALL	NO	NO		M_INS	
	6	DEL	DEL_ALL	NO	NO		M_DEL	
	7	RET	RET_ALL	NO	NO		M_RET	
RT_FUNC	1	USER_NM_RT	USER_NM	NO	NO			
	2	RT_NM	RT_NM	NO	NO			
	3	USER_NM_FUNC	USER_NM	NO	NO			
	4	FUNC_NM	FUNC_NM	NO	NO			
RT_SITE	1	USER_NM	USER_NM	NO	NO			
	2	RT_NM	RT_NM	NO	NO			
	3	SITE_NM	SITE_NM	NO	NO			
	4	RL_EXPR_LG	RL_EXPR	NO	YES	C_RL_EXPR_LG	M_RL_EXPR_LG	
	5	RL_EXPR_GL	RL_EXPR	NO	YES	C_RL_EXPR_GL	M_RL_EXPR_GL	
	6	IND_REPLICA	IND_REPLICA	NO	YES		M_IND_REPLICA	
	7	CR_DATE	DATE	NO	NO		M_RTS_CR_DATE	
	8	CR_TIME	TIME	NO	NO		M_RTS_CR_TIME	
	9	N_DROP	TURN_KEY_NR	NO	NO		M_N_DROP	
	10	N_DELETE	TURN_KEY_NR	NO	YES		M_N_DELETE	
	11	N_DELAY	TURN_KEY_NR	NO	NO		M_N_DELAY	
	12	DELAY_PERIOD	DELAY_PERIOD	NO	YES		M_DELAY_PERIOD	
	13	ROWCOUNT	ROW_COL_COUNT	NO	YES		M_ROWCOUNT	
RT_SYN	1	USER_NM_SYN	USER_NM	NO	NO			
	2	RT_NM_SYN	RT_NM	NO	NO			
	3	USER_NM_RT	USER_NM	NO	NO		M_SYN_USER_NM_RT	
	4	RT_NM	RT_NM	NO	NO		M_SYN_RT_NM	
R_TABLE	1	USER_NM	USER_NM	NO	NO			
	2	RT_NM	RT_NM	NO	NO			
	3	RT_TYPE	RT_TYPE	NO	NO		M_RT_TYPE	
	4	RL_EXPR	RL_EXPR	NO	YES	C_RL_EXPR_VIEW	M_RL_EXPR	
	5	INS_ALL	INS_ALL	NO	YES		M_RT_INS_ALL	
	6	DEL_ALL	DEL_ALL	NO	YES		M_RT_DEL_ALL	
	7	FUNC_NM_VW	FUNC_NM	YES	NO		M_FUNC_NM_VW	
SINGULAR	1	DUMMY	DUMMY	NO	NO			
	2	DOM_NM_FUNC	DOM_NM	NO	NO		M_DOM_NM_FUNC	
	3	DOM_NM_ARG	DOM_NM	NO	NO		M_DOM_NM_ARG	
	4	USER_NM_CATALOG	USER_NM	NO	NO		M_USER_NM_CATALOG	
SITE	1	SITE_NM	SITE_NM	NO	NO			
	2	IND_CATALOG_SITE	IND_CATALOG_SITE	NO	NO		M_IND_CATALOG_SITE	
TERMINAL	1	SITE_NM	SITE_NM	NO	NO			
	2	TERM_ID	TERM_ID	NO	NO			
TERM_DFLT	1	SITE_NM	SITE_NM	NO	NO			
	2	TERM_ID	TERM_ID	NO	NO			
	3	USER_NM	USER_NM	NO	NO			
	4	RT_NM	RT_NM	NO	NO			
	5	COL_NM	COL_NM	NO	NO			
	6	DEFAULT_VAL	DB_VAL	NO	NO		M_TDF_DEFAULT_VAL	
USER	1	USER_NM	USER_NM	NO	NO			
	2	USER_GRP_NM	USER_NM	YES	NO		M_USER_GRP_NM	
	3	IND_GDBA	IND_GDBA	NO	NO		M_IND_GDBA	

COLUMN (continued)

RT_NM	C#	COL_NM	DOM_NM	A	I	CA_NM_C	CA_NM_M	P
USER_SITE	1	USER_NM	USER_NM	NO	NO			
	2	SITE_NM	SITE_NM	NO	NO			
	3	TIME_BLOCK	TIME_BLOCK	NO	NO		M_UST_TIME_BLOCK	
	4	LOCK_QUOTA	LOCK_QUOTA	NO	NO		M_UST_LOCK_QUOTA	
	5	MAX_STOR_QUERY_ANT	STORAGE_USAGE	NO	NO		M_MAX_STOR_QUERY_ANT	
	6	MAX_STOR_QUERY_ACT	STORAGE_USAGE	NO	NO		M_MAX_STOR_QUERY_ACT	
	7	MAX_STOR_TOTAL_ANT	STORAGE_USAGE	NO	NO		M_MAX_STOR_TOTAL_ANT	
	8	MAX_STOR_TOTAL_ACT	STORAGE_USAGE	NO	NO		M_MAX_STOR_TOTAL_ACT	
	9	DBC_DOM	DBC_DOM	NO	NO		M_DBC_DOM	
	10	DBC_RT	DBC_RT	NO	NO		M_DBC_RT	
	11	DBC_COL	DBC_COL	NO	NO		M_DBC_COL	
	12	DBC_VIEW	DBC_VIEW	NO	NO		M_DBC_VIEW	
	13	DBC_CA_E	DBC_CA_E	NO	NO		M_DBC_CA_E	
	14	DBC_CA_R	DBC_CA_R	NO	NO		M_DBC_CA_R	
	15	DBC_CA_C	DBC_CA_C	NO	NO		M_DBC_CA_C	
	16	DBC_CA_D	DBC_CA_D	NO	NO		M_DBC_CA_D	
	17	DBC_CA_U	DBC_CA_U	NO	NO		M_DBC_CA_U	
	18	DBC_FUNC	DBC_FUNC	NO	NO		M_DBC_FUNC	
	19	DBC_ACCESS	DBC_ACCESS	NO	NO		M_DBC_ACCESS	
	20	DBC_DB_INDEX	DBC_DB_INDEX	NO	NO		M_DBC_DB_INDEX	
	21	DBC_FK	DBC_FK	NO	NO		M_DBC_FK	
	22	DBC_AUTH	DBC_AUTH	NO	NO		M_DBC_AUTH	
	23	DBC_SNAPSHOT	DBC_SNAPSHOT	NO	NO		M_DBC_SNAPSHOT	
	24	DBC_AUDIT_LOG	DBC_AUDIT_LOG	NO	NO		M_DBC_AUDIT_LOG	
	25	DBC_ARCH	DBC_ARCH	NO	NO		M_DBC_ARCH	
	26	DBC_DATE_ROUND	DBC_DATE_ROUND	NO	NO		M_DBC_DATE_ROUND	
	27	Q_A_MAYBE_ALL	Q_MAYBE_ALL	NO	NO		M_Q_A_MAYBE_ALL	
	28	Q_I_MAYBE_ALL	Q_MAYBE_ALL	NO	NO		M_Q_I_MAYBE_ALL	
	29	Q_MAYBE_ALL	Q_MAYBE_ALL	NO	NO		M_Q_MAYBE_ALL	
	30	Q_AR_ALL	Q_AR_ALL	NO	NO		M_Q_AR_ALL	
	31	Q_IR_ALL	Q_IR_ALL	NO	NO		M_Q_IR_ALL	
	32	Q_ESR_ALL	Q_ESR_ALL	NO	NO		M_Q_ESR_ALL	
	33	Q_ORDER_ALL	Q_ORDER_ALL	NO	NO		M_Q_ORDER_ALL	
	34	Q_ONCE_ALL	Q_ONCE_ALL	NO	NO		M_Q_ONCE_ALL	
	35	Q_DCO_ALL	Q_DCO_ALL	NO	NO		M_Q_DCO_ALL	
	36	Q_EXCL_SIBL_ALL	Q_EXCL_SIBL_ALL	NO	NO		M_Q_EXCL_SIBL_ALL	
	37	Q_DOD_ALL	Q_DOD_ALL	NO	NO		M_Q_DOD_ALL	
	38	Q_SAVE_ALL	Q_SAVE_ALL	NO	NO		M_Q_SAVE_ALL	
	39	Q_VALUE_ALL	Q_VALUE_ALL	NO	NO		M_Q_VALUE_ALL	
	40	IND_DBA_USER	IND_DBA_USER	NO	NO		M_IND_DBA_USER	
	41	GRANT_ALL	GRANT_ALL	NO	NO		M_GRANT_ALL	
	42	ASSIGN_ALL	ASSIGN_ALL	NO	NO		M_ASSIGN_ALL	

The DOMAIN data are retrieved by means of the following SQL statement:

```
SELECT DOM_NM, DOM_TYPE, COMP_ALL, BD_TYPE_NM, LENGTH, SCALE, CA_NM
FROM DOMAIN
WHERE USER_NM_DOM = 'RMV2'    /* username of catalog owner */
ORDER BY DOM_NM
```

DOMAIN

DOM_NM	DOM_TYPE	COM	BD_TYP	LENGTH	SCALE	CA_NM
ARCHIVE_NM	PRIMARY	YES	CHAR	30		D_ARCHIVE_NM
ARCH_ALL	SECUNDARY	NO	CHAR	3		
ARG_NM	PRIMARY	YES	CHAR	30		D_ARG_NM
ARG_SEQ#	PRIMARY	YES	NUMBER	4		
ASSIGN_ALL	SECUNDARY	NO	CHAR	3		
AUDIT_SEQ#	PRIMARY	YES	NUMBER	12		
BD_TYPE_NM	PRIMARY	YES	CHAR	30		D_BD_TYPE_NM
CA_DATE	PRIMARY	YES	DATE			D_CA_DATE
CA_NM	PRIMARY	YES	CHAR	30		D_CA_NM
CA_TIME	PRIMARY	YES	NUMBER	12	6	D_CA_TIME
CHECKOUT_ALL	SECUNDARY	NO	CHAR	3		

DOMAIN (continued)

DOM_NM	DOM_TYPE	COM	BD_TYP	LENGTH	SCALE	CA_NM
COL_NM	PRIMARY	YES	CHAR	30		D_COL_NM
COL_SEQ#	PRIMARY	YES	NUMBER	4		
COMP_ALL	SECUNDARY	NO	CHAR	3		
CONSTR_TYPE	SECUNDARY	NO	CHAR	9		
CS_SEQ#	PRIMARY	YES	NUMBER	2		
DATE	PRIMARY	YES	DATE			D_DATE
DBC_ACCESS	SECUNDARY	NO	CHAR	3		
DBC_ARCH	SECUNDARY	NO	CHAR	3		
DBC_AUDIT_LOG	SECUNDARY	NO	CHAR	3		
DBC_AUTH	SECUNDARY	NO	CHAR	3		
DBC_CA_C	SECUNDARY	NO	CHAR	3		
DBC_CA_D	SECUNDARY	NO	CHAR	3		
DBC_CA_E	SECUNDARY	NO	CHAR	3		
DBC_CA_R	SECUNDARY	NO	CHAR	3		
DBC_CA_U	SECUNDARY	NO	CHAR	3		
DBC_COL	SECUNDARY	NO	CHAR	3		
DBC_DATE_ROUND	SECUNDARY	NO	CHAR	3		
DBC_DB_INDEX	SECUNDARY	NO	CHAR	3		
DBC_DOM	SECUNDARY	NO	CHAR	3		
DBC_FK	SECUNDARY	NO	CHAR	3		
DBC_FUNC	SECUNDARY	NO	CHAR	3		
DBC_RT	SECUNDARY	NO	CHAR	3		
DBC_SNAPSHOT	SECUNDARY	NO	CHAR	3		
DBC_VIEW	SECUNDARY	NO	CHAR	3		
DB_VAL	SECUNDARY	YES	CHAR	9999		
DELAY_PERIOD	SECUNDARY	YES	NUMBER	3		
DEL_ALL	SECUNDARY	NO	CHAR	3		
DOM_NM	PRIMARY	YES	CHAR	30		D_DOM_NM
DOM_TYPE	SECUNDARY	NO	CHAR	9		
DR_SEQ#	PRIMARY	YES	NUMBER	4		
DS_SEQ#	PRIMARY	YES	NUMBER	2		
DUMMY	PRIMARY	NO	CHAR	1		D_DUMMY
FK_RULE	SECUNDARY	NO	CHAR	8		
FUNC_NM	PRIMARY	YES	CHAR	30		D_FUNC_NM
GRANT_ALL	SECUNDARY	NO	CHAR	3		
GROUP_ID	SECUNDARY	NO	CHAR	11		
HOST_LANG	SECUNDARY	YES	CHAR	10		
INDEX_NM	PRIMARY	YES	CHAR	30		D_INDEX_NM
IND_CATALOG_SITE	SECUNDARY	NO	CHAR	3		
IND_DBA_USER	SECUNDARY	NO	CHAR	3		
IND_DOD	SECUNDARY	NO	CHAR	3		
IND_GDBA	SECUNDARY	NO	CHAR	3		
IND_LENGTH	SECUNDARY	NO	CHAR	3		
IND_MARK	SECUNDARY	NO	CHAR	3		
IND_REPLICA	SECUNDARY	NO	CHAR	3		
IND_SCALE	SECUNDARY	NO	CHAR	3		
IND_STATIC	SECUNDARY	NO	CHAR	3		
IND_STORED	SECUNDARY	NO	CHAR	3		
IND_UNMARK	SECUNDARY	NO	CHAR	3		
INS_ALL	SECUNDARY	NO	CHAR	3		
INTERVAL	SECUNDARY	NO	NUMBER	8		
KEY_COL_SEQ#	PRIMARY	YES	NUMBER	4		
KEY_NM	PRIMARY	YES	CHAR	30		D_KEY_NM
KEY_TYPE	SECUNDARY	NO	CHAR	9		
LENGTH	SECUNDARY	YES	NUMBER	5		
LOCK_QUOTA	SECUNDARY	YES	NUMBER	6		
MARK_PREF	SECUNDARY	NO	CHAR	1		
PK_HIDDEN	SECUNDARY	NO	CHAR	3		
PK_UPD_ALL	SECUNDARY	NO	CHAR	3		
PROG_ID	PRIMARY	YES	CHAR	30		D_PROG_ID
Q_AR_ALL	SECUNDARY	NO	CHAR	3		
Q_DCO_ALL	SECUNDARY	NO	CHAR	3		
Q_DOD_ALL	SECUNDARY	NO	CHAR	3		
Q_ESR_ALL	SECUNDARY	NO	CHAR	3		
Q_EXCL_SIBL_ALL	SECUNDARY	NO	CHAR	3		
Q_IR_ALL	SECUNDARY	NO	CHAR	3		
Q_MAYBE_ALL	SECUNDARY	NO	CHAR	3		
Q_ONCE_ALL	SECUNDARY	NO	CHAR	3		
Q_ORDER_ALL	SECUNDARY	NO	CHAR	3		
Q_SAVE_ALL	SECUNDARY	NO	CHAR	3		
Q_VALUE_ALL	SECUNDARY	NO	CHAR	3		
RET_ALL	SECUNDARY	NO	CHAR	3		
RL_EXPR	SECUNDARY	NO	CHAR	9999		D_RL_EXPR
RL_KEYW	PRIMARY	YES	CHAR	30		D_RL_KEYW
ROUTINE_NM	PRIMARY	YES	CHAR	14		D_ROUTINE_NM
ROW_COL_COUNT	SECUNDARY	YES	NUMBER	12		

DOMAIN (continued)

DOM_NM	DOM_TYPE	CON	BD_TYP	LENGTH	SCALE	CA_NM
RT_NM	PRIMARY	YES	CHAR	30		D_RT_NM
RT_TYPE	SECUNDARY	NO	CHAR	8		
SCALE	SECUNDARY	YES	NUMBER	2		
SITE_NM	PRIMARY	YES	CHAR	30		D_SITE_NM
SNAPSHOT_ALL	SECUNDARY	NO	CHAR	3		
STORAGE_USAGE	SECUNDARY	YES	NUMBER	6		
TERM_ID	PRIMARY	YES	CHAR	30		D_TERM_ID
TIME	PRIMARY	YES	DATE			D_TIME
TIME_BLOCK	SECUNDARY	YES	NUMBER	6		
TTIME	SECUNDARY	NO	CHAR	2		
TURN_KEY_NR	SECUNDARY	YES	NUMBER	2		
UPD_ALL	SECUNDARY	NO	CHAR	3		
USER_NM	PRIMARY	YES	CHAR	30		D_USER_NM

The DOM_RANGE data are retrieved by means of the following SQL statement:

```
SELECT DOM_NM, DR_SEQ#, LOVAL, HIVAL
FROM DOM_RANGE
WHERE DOM_NM IN
    (SELECT DOM_NM
     FROM DOMAIN
     WHERE USER_NM_DOM = 'RMV2')  /* username of catalog owner */
ORDER BY DOM_NM, DR_SEQ#
```

DOM_RANGE

DOM_NM	DR_SEQ#	LOVAL	HIVAL
ARCH_ALL	1	NO	NO
	2	YES	YES
ARG_SEQ#	1	0001	9999
ASSIGN_ALL	1	NO	NO
	2	YES	YES
AUDIT_SEQ#	1	000000000001	999999999999
CHECKOUT_ALL	1	NO	NO
	2	YES	YES
COL_SEQ#	1	0001	9999
COMP_ALL	1	NO	NO
	2	YES	YES
CONSTR_TYPE	1	FD	FD
	2	INCL_DPCY	INCL_DPCY
	3	JD	JD
	4	MVD	MVD
CS_SEQ#	1	01	99
DBC_ACCESS	1	NO	NO
	2	YES	YES
DBC_ARCH	1	NO	NO
	2	YES	YES
DBC_AUDIT_LOG	1	NO	NO
	2	YES	YES
DBC_AUTH	1	NO	NO
	2	YES	YES
DBC_CA_C	1	NO	NO
	2	YES	YES
DBC_CA_D	1	NO	NO
	2	YES	YES
DBC_CA_E	1	NO	NO
	2	YES	YES
DBC_CA_R	1	NO	NO
	2	YES	YES
DBC_CA_U	1	NO	NO
	2	YES	YES
DBC_COL	1	NO	NO
	2	YES	YES
DBC_DATE_ROUND	1	NO	NO
	2	YES	YES
DBC_DB_INDEX	1	NO	NO
	2	YES	YES
DBC_DOM	1	NO	NO
	2	YES	YES
.....	..	.....	.....

DOM_RANGE (continued)

DOM_NM	DR_SEQ#	LOVAL	HIVAL
DBC_FK	1	NO	NO
	2	YES	YES
DBC_FUNC	1	NO	NO
	2	YES	YES
DBC_RT	1	NO	NO
	2	YES	YES
DBC_SNAPSHOT	1	NO	NO
	2	YES	YES
DBC_VIEW	1	NO	NO
	2	YES	YES
DB_VAL	1	NO	NO
	2	YES	YES
DELAY_PERIOD	1	001	999
DEL_ALL	1	NO	NO
	2	YES	YES
DOM_TYPE	1	PRIMARY	PRIMARY
	2	SECONDARY	SECONDARY
DR_SEQ#	1	0001	9999
DS_SEQ#	1	01	99
FK_RULE	1	CASCADE	CASCADE
	2	MARK	MARK
	3	RESTRICT	RESTRICT
GRANT_ALL	1	NO	NO
	2	YES	YES
IND_CATALOG_SITE	1	NO	NO
	2	YES	YES
IND_DBA_USER	1	NO	NO
	2	YES	YES
IND_DOD	1	NO	NO
	2	YES	YES
IND_GDBA	1	NO	NO
	2	YES	YES
IND_LENGTH	1	NO	NO
	2	YES	YES
IND_MARK	1	NO	NO
	2	YES	YES
IND_REPLICA	1	NO	NO
	2	YES	YES
IND_SCALE	1	NO	NO
	2	YES	YES
IND_STATIC	1	NO	NO
	2	YES	YES
IND_STORED	1	NO	NO
	2	YES	YES
IND_UNMARK	1	NO	NO
	2	YES	YES
INS_ALL	1	NO	NO
	2	YES	YES
INTERVAL	1	00000001	99999999
KEY_COL_SEQ#	1	0001	9999
KEY_TYPE	1	ALTERNATE	ALTERNATE
	2	FOREIGN	FOREIGN
	3	INDEX	INDEX
	4	PRIMARY	PRIMARY
LENGTH	1	00001	99999
LOCK_QUOTA	1	000001	999999
MARK_PREF	1	A	A
	2	I	I
PK_HIDDEN	1	NO	NO
	2	YES	YES
PK_UPD_ALL	1	NO	NO
	2	YES	YES
Q_AR_ALL	1	NO	NO
	2	YES	YES
Q_DCO_ALL	1	NO	NO
	2	YES	YES
Q_DOD_ALL	1	NO	NO
	2	YES	YES
Q_ESR_ALL	1	NO	NO
	2	YES	YES
Q_EXCL_SIBL_ALL	1	NO	NO
	2	YES	YES
Q_IR_ALL	1	NO	NO
	2	YES	YES
.....	..	.....	.....

DOM_RANGE (continued)

DOM_NM	DR_SEQ#	LOVAL	HIVAL
Q_MAYBE_ALL	1	NO	NO
	2	YES	YES
Q_ONCE_ALL	1	NO	NO
	2	YES	YES
Q_ORDER_ALL	1	NO	NO
	2	YES	YES
Q_SAVE_ALL	1	NO	NO
	2	YES	YES
Q_VALUE_ALL	1	NO	NO
	2	YES	YES
RET_ALL	1	NO	NO
	2	YES	YES
ROW_COL_COUNT	1	000000000000	999999999999
RT_TYPE	1	BASE	BASE
	2	CHECKOUT	CHECKOUT
	3	REL_ASGN	REL_ASGN
	4	SNAPSHOT	SNAPSHOT
	5	VIEW	VIEW
SCALE	1	00	99
SNAPSHOT_ALL	1	NO	NO
	2	YES	YES
STORAGE_USAGE	1	000001	999999
TIME_BLOCK	1	000001	999999
TTIME	1	TC	TC
	2	TT	TT
TURN_KEY_NR	1	01	99
UPD_ALL	1	NO	NO
	2	YES	YES

The KEY data are retrieved by means of the following SQL statement:

```
SELECT RT_NM, KEY_NM, KEY_TYPE, CA_NM, UPD_ACT, DEL_ACT
FROM KEY
WHERE USER_NM = 'RMV2'    /* username of catalog owner */
ORDER BY RT_NM, KEY_NM
```

KEY

RT_NM	KEY_NM	KEY_TYPE	CA_NM	UPD_ACT	DEL_ACT
ARCHIVE	ARCHIVE_RTSITE	FOREIGN	ARCHIVE_RTSITE	CASCADE	CASCADE
	PRIMARY_KEY	PRIMARY	PK_ARCHIVE		
ARGUMENT	ALTERNATE_ARG_NM	ALTERNATE	ALTERNATE_ARG_NM		
	ARGUMENT_FUNCTION	FOREIGN	ARGUMENT_FUNCTION	CASCADE	CASCADE
	PRIMARY_KEY	PRIMARY	PK_ARGUMENT		
AUDIT_INTERVAL	AUDITINTERVAL_RTABLE	FOREIGN	AUDITINTERVAL_RTABLE	CASCADE	CASCADE
	AUDITINTERVAL_USER_OFF	FOREIGN	AUDITINTERVAL_USER_OFF	CASCADE	CASCADE
	AUDITINTERVAL_USER_ON	FOREIGN	AUDITINTERVAL_USER_ON	CASCADE	CASCADE
	PRIMARY_KEY	PRIMARY	PK_AUDIT_INTERVAL		
AUDIT_LOG	AUDITLOG_COLUMN	FOREIGN	AUDITLOG_COLUMN	CASCADE	CASCADE
	AUDITLOG_PROGRAM	FOREIGN	AUDITLOG_PROGRAM	CASCADE	RESTRICT
	AUDITLOG_TERMINAL	FOREIGN	AUDITLOG_TERMINAL	CASCADE	RESTRICT
	AUDITLOG_USER	FOREIGN	AUDITLOG_USER	CASCADE	RESTRICT
	PRIMARY_KEY	PRIMARY	PK_AUDIT_LOG		
BASIC_DATATYPE	PRIMARY_KEY	PRIMARY	PK_BASIC_DATATYPE		
CA_SITE	CASITE_CONDACT	FOREIGN	CASITE_CONDACT	CASCADE	CASCADE
	CASITE_SITE	FOREIGN	CASITE_SITE	CASCADE	CASCADE
	PRIMARY_KEY	PRIMARY	PK_CA_SITE		
COLUMN	ALTERNATE_CA_C	ALTERNATE	ALTERNATE_CA_C		
	ALTERNATE_CA_M	ALTERNATE	ALTERNATE_CA_M		
	ALTERNATE_SEQ#	ALTERNATE	ALTERNATE_SEQ#_COL		
	COLUMN_BASICDATATYPE	FOREIGN	COLUMN_BASICDATATYPE	RESTRICT	RESTRICT
	COLUMN_COLUMN	FOREIGN	COLUMN_COLUMN	CASCADE	CASCADE
	COLUMN_CONDACT_COLCONSTR	FOREIGN	COLUMN_CONDACT_COLCONSTR	CASCADE	NULLIFY
	COLUMN_CONDACT_MARK	FOREIGN	COLUMN_CONDACT_MARK	CASCADE	NULLIFY
	COLUMN_DOMAIN	FOREIGN	COLUMN_DOMAIN	CASCADE	RESTRICT
	COLUMN_RTABLE	FOREIGN	COLUMN_RTABLE	CASCADE	CASCADE
	PRIMARY_KEY	PRIMARY	PK_COLUMN		
.....	.....	.....	.....	.....	.....

KEY (continued)

RT_NM	KEY_NM	KEY_TYPE	CA_NM	UPD_ACT	DEL_ACT
COLUMN_STRUCT	ALTERNATE_SEQ#	ALTERNATE	ALTERNATE_SEQ#_CS		
	COLSTRUCT_COLUMN_SUB	FOREIGN	COLSTRUCT_COLUMN_SUB	CASCADE	RESTRICT
	COLSTRUCT_COLUMN_SUP	FOREIGN	COLSTRUCT_COLUMN_SUP	CASCADE	CASCADE
	PRIMARY_KEY	PRIMARY	PK_COLUMN_STRUCT		
COL_AUTH	COLAUTH_COLUMN	FOREIGN	COLAUTH_COLUMN	CASCADE	CASCADE
	COLAUTH_RTAUTH	FOREIGN	COLAUTH_RTAUTH	CASCADE	CASCADE
	PRIMARY_KEY	PRIMARY	PK_COL_AUTH		
COL_CA	COLCA_COLUMN	FOREIGN	COLCA_COLUMN	CASCADE	CASCADE
	COLCA_CONDACT	FOREIGN	COLCA_CONDACT	CASCADE	CASCADE
	PRIMARY_KEY	PRIMARY	PK_COL_CA		
COL_SITE	COLSITE_COLUMN	FOREIGN	COLSITE_COLUMN	CASCADE	CASCADE
	COLSITE_RTSITE	FOREIGN	COLSITE_RTSITE	CASCADE	CASCADE
	PRIMARY_KEY	PRIMARY	PK_COL_SITE		
COND_ACT	CONDACT_USER	FOREIGN	CONDACT_USER	CASCADE	CASCADE
	PRIMARY_KEY	PRIMARY	PK_COND_ACT		
DB_INDEX	DBINDEX_DOMAIN	FOREIGN	DBINDEX_DOMAIN	CASCADE	CASCADE
	PRIMARY_KEY	PRIMARY	PK_DB_INDEX		
DB_INDEX_COL	DBINDEXCOL_COLUMN	FOREIGN	DBINDEXCOL_COLUMN	CASCADE	CASCADE
	DBINDEXCOL_DBINDEX	FOREIGN	DBINDEXCOL_DBINDEX	CASCADE	CASCADE
	PRIMARY_KEY	PRIMARY	PK_DB_INDEX_COL		
DOMAIN	ALTERNATE_CA_NM	ALTERNATE	ALTERNATE_CA_NM_D		
	DOMAIN_BASICDATATYPE	FOREIGN	DOMAIN_BASICDATATYPE	RESTRICT	RESTRICT
	DOMAIN_CONDACT	FOREIGN	DOMAIN_CONDACT	CASCADE	NULLIFY
	DOMAIN_USER	FOREIGN	DOMAIN_USER	CASCADE	RESTRICT
	PRIMARY_KEY	PRIMARY	PK_DOMAIN		
DOMAIN_STRUCT	ALTERNATE_SEQ#	ALTERNATE	ALTERNATE_SEQ#_DS		
	DOMSTRUCT_DOMAIN_SUB	FOREIGN	DOMSTRUCT_DOMAIN_SUB	CASCADE	RESTRICT
	DOMSTRUCT_DOMAIN_SUP	FOREIGN	DOMSTRUCT_DOMAIN_SUP	CASCADE	CASCADE
	PRIMARY_KEY	PRIMARY	PK_DOMAIN_STRUCT		
DOM_RANGE	DOMRANGE_DOMAIN	FOREIGN	DOMRANGE_DOMAIN	CASCADE	CASCADE
	PRIMARY_KEY	PRIMARY	PK_DOM_RANGE		
FUNCTION	ALTERNATE_ROUT_EXEC	ALTERNATE	ALTERNATE_ROUT_EXEC		
	ALTERNATE_ROUT_SRC	ALTERNATE	ALTERNATE_ROUT_SRC		
	FUNCTION_FUNCTION_DOD	FOREIGN	FUNCTION_FUNCTION_DOD	CASCADE	NULLIFY
	FUNCTION_FUNCTION_INV	FOREIGN	FUNCTION_FUNCTION_INV	CASCADE	NULLIFY
	FUNCTION_USER	FOREIGN	FUNCTION_USER	CASCADE	CASCADE
	PRIMARY_KEY	PRIMARY	PK_FUNCTION		
FUNC_SITE	FUNCSITE_FUNCTION	FOREIGN	FUNCSITE_FUNCTION	CASCADE	CASCADE
	FUNCSITE_SITE	FOREIGN	FUNCSITE_SITE	CASCADE	CASCADE
	PRIMARY_KEY	PRIMARY	PK_FUNC_SITE		
KEY	ALTERNATE_CA_NM	ALTERNATE	ALTERNATE_CA_NM_K		
	ALTERNATE_INDEX	ALTERNATE	ALTERNATE_INDEX		
	KEY_CONDACT	FOREIGN	KEY_CONDACT	CASCADE	RESTRICT
	KEY_RTABLE	FOREIGN	KEY_RTABLE	CASCADE	CASCADE
	PRIMARY_KEY	PRIMARY	PK_KEY		
KEY_COLUMN	ALTERNATE_SEQ#	ALTERNATE	ALTERNATE_SEQ#_KC		
	KEYCOLUMN_COLUMN	FOREIGN	KEYCOLUMN_COLUMN	CASCADE	CASCADE
	KEYCOLUMN_KEY	FOREIGN	KEYCOLUMN_KEY	CASCADE	CASCADE
	PRIMARY_KEY	PRIMARY	PK_KEY_COLUMN		
KEY_REF	KEYREF_KEY	FOREIGN	KEYREF_KEY	CASCADE	CASCADE
	KEYREF_RTABLE	FOREIGN	KEYREF_RTABLE	CASCADE	CASCADE
	PRIMARY_KEY	PRIMARY	PK_KEY_REF		
LOG_PK_COLUMN	LOGPKCOLUMN_AUDITLOG	FOREIGN	LOGPKCOLUMN_AUDITLOG	CASCADE	RESTRICT
	LOGPKCOLUMN_COLUMN	FOREIGN	LOGPKCOLUMN_COLUMN	CASCADE	CASCADE
	PRIMARY_KEY	PRIMARY	PK_LOG_PK_COLUMN		
PROGRAM	PRIMARY_KEY	PRIMARY	PK_PROGRAM		
	PROGRAM_SITE	FOREIGN	PROGRAM_SITE	CASCADE	CASCADE
PROG_DFLT	PRIMARY_KEY	PRIMARY	PK_PROG_DFLT		
	PROGDFLT_COLUMN	FOREIGN	PROGDFLT_COLUMN	CASCADE	CASCADE
	PROGDFLT_PROGRAM	FOREIGN	PROGDFLT_PROGRAM	CASCADE	CASCADE
RL_KEYWORD	PRIMARY_KEY	PRIMARY	PK_RL_KEYWORD		
RT_AUTH	PRIMARY_KEY	PRIMARY	PK_RT_AUTH		
	RTAUTH_RTABLE	FOREIGN	RTAUTH_RTABLE	CASCADE	CASCADE
	RTAUTH_USER_GRANTEE	FOREIGN	RTAUTH_USER_GRANTEE	CASCADE	CASCADE
	RTAUTH_USER_GRANTOR	FOREIGN	RTAUTH_USER_GRANTOR	CASCADE	CASCADE
RT_CA	PRIMARY_KEY	PRIMARY	PK_RT_CA		
	RTCA_CONDACT	FOREIGN	RTCA_CONDACT	CASCADE	CASCADE
	RTCA_RTABLE	FOREIGN	RTCA_RTABLE	CASCADE	CASCADE
RT_FUNC	PRIMARY_KEY	PRIMARY	PK_RT_FUNC		
	RTFUNC_FUNCTION	FOREIGN	RTFUNC_FUNCTION	CASCADE	CASCADE
	RTFUNC_RTABLE	FOREIGN	RTFUNC_RTABLE	CASCADE	CASCADE
RT_SITE	PRIMARY_KEY	PRIMARY	PK_RT_SITE		
	RTSITE_RTABLE	FOREIGN	RTSITE_RTABLE	CASCADE	CASCADE
	RTSITE_SITE	FOREIGN	RTSITE_SITE	CASCADE	CASCADE
RT_SYN	PRIMARY_KEY	PRIMARY	PK_RT_SYN		
	RTSYN_RTABLE	FOREIGN	RTSYN_RTABLE	CASCADE	CASCADE

KEY (continued)

RT_NM	KEY_NM	KEY_TYPE	CA_NM	UPD_ACT	DEL_ACT
R_TABLE	PRIMARY_KEY	PRIMARY	PK_R_TABLE		
	RTABLE_FUNCTION	FOREIGN	RTABLE_FUNCTION	CASCADE	NULLIFY
	RTABLE_USER	FOREIGN	RTABLE_USER	CASCADE	CASCADE
SINGULAR	PRIMARY_KEY	PRIMARY	PK_SINGULAR		
	SINGULAR_DOMAIN_ARG	FOREIGN	SINGULAR_DOMAIN_ARG	CASCADE	CASCADE
	SINGULAR_DOMAIN_FUNC	FOREIGN	SINGULAR_DOMAIN_FUNC	CASCADE	RESTRICT
	SINGULAR_USER	FOREIGN	SINGULAR_USER	CASCADE	RESTRICT
SITE	PRIMARY_KEY	PRIMARY	PK_SITE		
TERMINAL	PRIMARY_KEY	PRIMARY	PK_TERMINAL		
	TERMINAL_SITE	FOREIGN	TERMINAL_SITE	CASCADE	CASCADE
TERM_DFLT	PRIMARY_KEY	PRIMARY	PK_TERM_DFLT		
	TERMDFLT_COLUMN	FOREIGN	TERMDFLT_COLUMN	CASCADE	CASCADE
	TERMDFLT_TERMINAL	FOREIGN	TERMDFLT_TERMINAL	CASCADE	CASCADE
USER	PRIMARY_KEY	PRIMARY	PK_USER		
	USER_USER	FOREIGN	USER_USER	CASCADE	RESTRICT
USER_SITE	PRIMARY_KEY	PRIMARY	PK_USER_SITE		
	USERSITE_SITE	FOREIGN	USERSITE_SITE	CASCADE	CASCADE
	USERSITE_USER	FOREIGN	USERSITE_USER	CASCADE	CASCADE

The KEY_REF data are retrieved by means of the following SQL statement:

```
SELECT RT_NM_FK, KEY_NM, RT_NM_PK
FROM KEY_REF
WHERE USER_NM_PK = 'RMV2'    /* username of catalog owner */
  AND USER_NM_FK = 'RMV2'
ORDER BY RT_NM_FK, KEY_NM
```

KEY_REF

RT_NM_FK	KEY_NM	RT_NM_PK
ARCHIVE	ARCHIVE_RTSITE	RT_SITE
ARGUMENT	ARGUMENT_FUNCTION	FUNCTION
AUDIT_INTERVAL	AUDITINTERVAL_RTABLE	R_TABLE
	AUDITINTERVAL_USER_OFF	USER
	AUDITINTERVAL_USER_ON	USER
AUDIT_LOG	AUDITLOG_COLUMN	COLUMN
	AUDITLOG_PROGRAM	PROGRAM
	AUDITLOG_TERMINAL	TERMINAL
	AUDITLOG_USER	USER
CA_SITE	CASITE_CONDACT	COND_ACT
	CASITE_SITE	SITE
COLUMN	COLUMN_BASICDATATYPE	BASIC_DATATYPE
	COLUMN_COLUMN	COLUMN
	COLUMN_CONDACT_COLCONSTR	COND_ACT
	COLUMN_CONDACT_MARK	COND_ACT
	COLUMN_DOMAIN	DOMAIN
	COLUMN_RTABLE	R_TABLE
COLUMN_STRUCT	COLSTRUCT_COLUMN_SUB	COLUMN
	COLSTRUCT_COLUMN_SUP	COLUMN
COL_AUTH	COLAUTH_COLUMN	COLUMN
	COLAUTH_RTAUTH	RT_AUTH
COL_CA	COLCA_COLUMN	COLUMN
	COLCA_CONDACT	COND_ACT
COL_SITE	COLSITE_COLUMN	COLUMN
	COLSITE_RTSITE	RT_SITE
COND_ACT	CONDACT_USER	USER
DB_INDEX	DBINDEX_DOMAIN	DOMAIN
DB_INDEX_COL	DBINDEXCOL_COLUMN	COLUMN
	DBINDEXCOL_DBINDEX	DB_INDEX
DOMAIN	DOMAIN_BASICDATATYPE	BASIC_DATATYPE
	DOMAIN_CONDACT	COND_ACT
	DOMAIN_USER	USER
DOMAIN_STRUCT	DOMSTRUCT_DOMAIN_SUB	DOMAIN
	DOMSTRUCT_DOMAIN_SUP	DOMAIN
DOM_RANGE	DOMRANGE_DOMAIN	DOMAIN
FUNCTION	FUNCTION_FUNCTION_DOD	FUNCTION
	FUNCTION_FUNCTION_INV	FUNCTION
	FUNCTION_USER	USER

KEY_REF (continued)

RT_NM_FK	KEY_NM	RT_NM_PK
FUNC_SITE	FUNCSITE_FUNCTION	FUNCTION
	FUNCSITE_SITE	SITE
KEY	KEY_CONDACT	COND_ACT
	KEY_RTABLE	R_TABLE
KEY_COLUMN	KEYCOLUMN_COLUMN	COLUMN
	KEYCOLUMN_KEY	KEY
KEY_REF	KEYREF_KEY	KEY
	KEYREF_RTABLE	R_TABLE
LOG_PK_COLUMN	LOGPKCOLUMN_AUDITLOG	AUDIT_LOG
	LOGPKCOLUMN_COLUMN	COLUMN
PROGRAM	PROGRAM_SITE	SITE
PROG_DFLT	PROGDFLT_COLUMN	COLUMN
	PROGDFLT_PROGRAM	PROGRAM
RT_AUTH	RTAUTH_RTABLE	R_TABLE
	RTAUTH_USER_GRANTEE	USER
	RTAUTH_USER_GRANTOR	USER
RT_CA	RTCA_CONDACT	COND_ACT
	RTCA_RTABLE	R_TABLE
RT_FUNC	RTFUNC_FUNCTION	FUNCTION
	RTFUNC_RTABLE	R_TABLE
RT_SITE	RTSITE_RTABLE	R_TABLE
	RTSITE_SITE	SITE
RT_SYN	RTSYN_RTABLE	R_TABLE
R_TABLE	RTABLE_FUNCTION	FUNCTION
	RTABLE_USER	USER
SINGULAR	SINGULAR_DOMAIN_ARG	DOMAIN
	SINGULAR_DOMAIN_FUNC	DOMAIN
	SINGULAR_USER	USER
TERMINAL	TERMINAL_SITE	SITE
TERM_DFLT	TERMDFLT_COLUMN	COLUMN
	TERMDFLT_TERMINAL	TERMINAL
USER	USER_USER	USER
USER_SITE	USERSITE_SITE	SITE
	USERSITE_USER	USER

The KEY_COLUMN data are retrieved by means of the following SQL statement:

```
SELECT RT_NM, KEY_NM, KEY_COL_SEQ#, COL_NM
FROM KEY_COLUMN
WHERE USER_NM  = 'RMV2'   /* username of catalog owner */
ORDER BY RT_NM, KEY_NM, KEY_COL_SEQ#
```

KEY_COLUMN

RT_NM	KEY_NM	KEY_COL_SEQ#	COL_NM
ARCHIVE	ARCHIVE_RTSITE	1	USER_NM
		2	RT_NM
		3	SITE_NM
	PRIMARY_KEY	1	USER_NM
		2	ARCHIVE_NM
ARGUMENT	ALTERNATE_ARG_NM	1	USER_NM
		2	FUNC_NM
		3	ARG_NM
	ARGUMENT_FUNCTION	1	USER_NM
		2	FUNC_NM
	PRIMARY_KEY	1	USER_NM
		2	FUNC_NM
		3	ARG_SEQ#
AUDIT_INTERVAL	AUDITINTERVAL_RTABLE	1	USER_NM
		2	RT_NM
	AUDITINTERVAL_USER_OFF	1	USER_NM_LOG_OFF
	AUDITINTERVAL_USER_ON	1	USER_NM_LOG_ON
	PRIMARY_KEY	1	USER_NM
		2	RT_NM
		3	DATE_LOG_ON
		4	TIME_LOG_ON
.....	.....	..	.....

KEY_COLUMN (continued)

RT_NM	KEY_NM	KEY_COL_SEQ#	COL_NM
AUDIT_LOG	AUDITLOG_COLUMN	1	USER_NM_RT
		2	RT_NM
		3	COL_NM
	AUDITLOG_PROGRAM	1	SITE_NM
		2	PROG_ID
	AUDITLOG_TERMINAL	1	SITE_NM
		2	TERM_ID
	AUDITLOG_USER	1	USER_NM_MUT
	PRIMARY_KEY	1	AUDIT_SEQ#
BASIC_DATATYPE	PRIMARY_KEY	1	BD_TYPE_NM
CA_SITE	CASITE_CONDACT	1	USER_NM
		2	CA_NM
	CASITE_SITE	1	SITE_NM
	PRIMARY_KEY	1	USER_NM
		2	CA_NM
		3	SITE_NM
COLUMN	ALTERNATE_CA_C	1	USER_NM
		2	CA_NM_C
	ALTERNATE_CA_M	1	USER_NM
		2	CA_NM_M
	ALTERNATE_SEQ#	1	USER_NM
		2	RT_NM
		3	COL_SEQ#
	COLUMN_BASICDATATYPE	1	BD_TYPE_NM
	COLUMN_COLUMN	1	USER_NM_BASE
		2	RT_NM_BASE
		3	COL_NM_BASE
	COLUMN_CONDACT_COLCONSTR	1	USER_NM
		2	CA_NM_C
	COLUMN_CONDACT_MARK	1	USER_NM
		2	CA_NM_M
	COLUMN_DOMAIN	1	DOM_NM
	COLUMN_RTABLE	1	USER_NM
		2	RT_NM
	PRIMARY_KEY	1	USER_NM
		2	RT_NM
		3	COL_NM
COLUMN_STRUCT	ALTERNATE_SEQ#	1	USER_NM
		2	RT_NM
		3	COL_NM_SUP
		4	CS_SEQ#
	COLSTRUCT_COLUMN_SUB	1	USER_NM
		2	RT_NM
		3	COL_NM_SUB
	COLSTRUCT_COLUMN_SUP	1	USER_NM
		2	RT_NM
		3	COL_NM_SUP
	PRIMARY_KEY	1	USER_NM
		2	RT_NM
		3	COL_NM_SUP
		4	COL_NM_SUB
COL_AUTH	COLAUTH_COLUMN	1	USER_NM_RT
		2	RT_NM
		3	COL_NM
	COLAUTH_RTAUTH	1	USER_NM_RT
		2	RT_NM
		3	USER_NM_GRANTOR
		4	USER_NM_GRANTEE
	PRIMARY_KEY	1	USER_NM_RT
		2	RT_NM
		3	COL_NM
		4	USER_NM_GRANTOR
		5	USER_NM_GRANTEE
COL_CA	COLCA_COLUMN	1	USER_NM_RT
		2	RT_NM
		3	COL_NM
	COLCA_CONDACT	1	USER_NM_CA
		2	CA_NM
	PRIMARY_KEY	1	USER_NM_RT
		2	RT_NM
		3	COL_NM
		4	USER_NM_CA
		5	CA_NM
.....	.....	..	.....

KEY_COLUMN (continued)

RT_NM	KEY_NM	KEY_COL_SEQ#	COL_NM
COL_SITE	COLSITE_COLUMN	1	USER_NM
		2	RT_NM
		3	COL_NM
	COLSITE_RTSITE	1	USER_NM
		2	RT_NM
		3	SITE_NM
	PRIMARY_KEY	1	USER_NM
		2	RT_NM
		3	COL_NM
		4	SITE_NM
COND_ACT	CONDACT_USER	1	USER_NM
	PRIMARY_KEY	1	USER_NM
		2	CA_NM
DB_INDEX	DBINDEX_DOMAIN	1	DOM_NM
	PRIMARY_KEY	1	USER_NM
		2	INDEX_NM
DB_INDEX_COL	DBINDEXCOL_COLUMN	1	USER_NM_RT
		2	RT_NM
		3	COL_NM
	DBINDEXCOL_DBINDEX	1	USER_NM_DBI
		2	INDEX_NM
	PRIMARY_KEY	1	USER_NM_DBI
		2	INDEX_NM
		3	USER_NM_RT
		4	RT_NM
		5	COL_NM
DOMAIN	ALTERNATE_CA_NM	1	USER_NM_CA
		2	CA_NM
	DOMAIN_BASICDATATYPE	1	BD_TYPE_NM
	DOMAIN_CONDACT	1	USER_NM_CA
		2	CA_NM
	DOMAIN_USER	1	USER_NM_DOM
	PRIMARY_KEY	1	DOM_NM
DOMAIN_STRUCT	ALTERNATE_SEQ#	1	DOM_NM_SUP
		2	DOM_NM_SUB
		3	DS_SEQ#
	DOMSTRUCT_DOMAIN_SUB	1	DOM_NM_SUB
	DOMSTRUCT_DOMAIN_SUP	1	DOM_NM_SUP
	PRIMARY_KEY	1	DOM_NM_SUP
		2	DOM_NM_SUB
DOM_RANGE	DOMRANGE_DOMAIN	1	DOM_NM
	PRIMARY_KEY	1	DOM_NM
		2	DR_SEQ#
FUNCTION	ALTERNATE_ROUT_EXEC	1	USER_NM
		2	ROUT_EXEC
	ALTERNATE_ROUT_SRC	1	USER_NM
		2	ROUT_SRC
	FUNCTION_FUNCTION_DOD	1	USER_NM
		2	FUNC_NM_DOD
	FUNCTION_FUNCTION_INV	1	USER_NM
		2	FUNC_NM_INV
	FUNCTION_USER	1	USER_NM
	PRIMARY_KEY	1	USER_NM
		2	FUNC_NM
FUNC_SITE	FUNCSITE_FUNCTION	1	USER_NM
		2	FUNC_NM
	FUNCSITE_SITE	1	SITE_NM
	PRIMARY_KEY	1	USER_NM
		2	FUNC_NM
		3	SITE_NM
KEY	ALTERNATE_CA_NM	1	USER_NM
		2	CA_NM
	ALTERNATE_INDEX	1	INDEX_NM
	KEY_CONDACT	1	USER_NM
		2	CA_NM
	KEY_RTABLE	1	USER_NM
		2	RT_NM
	PRIMARY_KEY	1	USER_NM
		2	RT_NM
		3	KEY_NM
.....	.....	..	.....

KEY_COLUMN (continued)

RT_NM	KEY_NM	KEY_COL_SEQ#	COL_NM
KEY_COLUMN	ALTERNATE_SEQ#	1	USER_NM
		2	RT_NM
		3	KEY_NM
		4	KEY_COL_SEQ#
	KEYCOLUMN_COLUMN	1	USER_NM
		2	RT_NM
		3	COL_NM
	KEYCOLUMN_KEY	1	USER_NM
		2	RT_NM
		3	KEY_NM
	PRIMARY_KEY	1	USER_NM
		2	RT_NM
		3	KEY_NM
		4	COL_NM
KEY_REF	KEYREF_KEY	1	USER_NM_FK
		2	RT_NM_FK
		3	KEY_NM
	KEYREF_RTABLE	1	USER_NM_PK
		2	RT_NM_PK
	PRIMARY_KEY	1	USER_NM_PK
		2	RT_NM_PK
		3	USER_NM_FK
		4	RT_NM_FK
		5	KEY_NM
LOG_PK_COLUMN	LOGPKCOLUMN_AUDITLOG	1	AUDIT_SEQ#
	LOGPKCOLUMN_COLUMN	1	USER_NM
		2	RT_NM
		3	COL_NM
	PRIMARY_KEY	1	AUDIT_SEQ#
		2	USER_NM
		3	RT_NM
		4	COL_NM
PROGRAM	PRIMARY_KEY	1	SITE_NM
		2	PROG_ID
	PROGRAM_SITE	1	SITE_NM
PROG_DFLT	PRIMARY_KEY	1	SITE_NM
		2	PROG_ID
		3	USER_NM
		4	RT_NM
		5	COL_NM
	PROGDFLT_COLUMN	1	USER_NM
		2	RT_NM
		3	COL_NM
	PROGDFLT_PROGRAM	1	SITE_NM
		2	PROG_ID
RL_KEYWORD	PRIMARY_KEY	1	RL_KEYW
RT_AUTH	PRIMARY_KEY	1	USER_NM_RT
		2	RT_NM
		3	USER_NM_GRANTOR
		4	USER_NM_GRANTEE
	RTAUTH_RTABLE	1	USER_NM_RT
		2	RT_NM
	RTAUTH_USER_GRANTEE	1	USER_NM_GRANTEE
	RTAUTH_USER_GRANTOR	1	USER_NM_GRANTOR
RT_CA	PRIMARY_KEY	1	USER_NM_RT
		2	RT_NM
		3	USER_NM_CA
		4	CA_NM
	RTCA_CONDACT	1	USER_NM_CA
		2	CA_NM
	RTCA_RTABLE	1	USER_NM_RT
		2	RT_NM
RT_FUNC	PRIMARY_KEY	1	USER_NM_RT
		2	RT_NM
		3	USER_NM_FUNC
		4	FUNC_NM
	RTFUNC_FUNCTION	1	USER_NM_FUNC
		2	FUNC_NM
	RTFUNC_RTABLE	1	USER_NM_RT
		2	RT_NM

KEY_COLUMN (continued)

RT_NM	KEY_NM	KEY_COL_SEQ#	COL_NM
RT_SITE	PRIMARY_KEY	1	USER_NM
		2	RT_NM
		3	SITE_NM
	RTSITE_RTABLE	1	USER_NM
		2	RT_NM
	RTSITE_SITE	1	SITE_NM
RT_SYN	PRIMARY_KEY	1	USER_NM_SYN
		2	RT_NM_SYN
	RTSYN_RTABLE	1	USER_NM_RT
		2	RT_NM
R_TABLE	PRIMARY_KEY	1	USER_NM
		2	RT_NM
	RTABLE_FUNCTION	1	USER_NM
		2	FUNC_NM_VW
	RTABLE_USER	1	USER_NM
SINGULAR	PRIMARY_KEY	1	DUMMY
	SINGULAR_DOMAIN_ARG	1	DOM_NM_ARG
	SINGULAR_DOMAIN_FUNC	1	DOM_NM_FUNC
	SINGULAR_USER	1	USER_NM_CATALOG
SITE	PRIMARY_KEY	1	SITE_NM
TERMINAL	PRIMARY_KEY	1	SITE_NM
		2	TERM_ID
	TERMINAL_SITE	1	SITE_NM
TERM_DFLT	PRIMARY_KEY	1	SITE_NM
		2	TERM_ID
		3	USER_NM
		4	RT_NM
		5	COL_NM
	TERMDFLT_COLUMN	1	USER_NM
		2	RT_NM
		3	COL_NM
	TERMDFLT_TERMINAL	1	SITE_NM
		2	TERM_ID
USER	PRIMARY_KEY	1	USER_NM
	USER_USER	1	USER_GRP_NM
USER_SITE	PRIMARY_KEY	1	USER_NM
		2	SITE_NM
	USERSITE_SITE	1	SITE_NM
	USERSITE_USER	1	USER_NM

The COND_ACT data are retrieved by means of the following SQL statement:

```
SELECT CA_NM, TTIME, CONSTR_TYPE
FROM COND_ACT
WHERE USER_NM = 'RMV2'   /* username of catalog owner */
  AND (
          (USER_NM, CA_NM) IN
               (SELECT USER_NM_CA, CA_NM
                FROM RT_CA
               )
        OR
          (USER_NM, CA_NM) IN
               (SELECT USER_NM_CA, CA_NM
                FROM COL_CA
               )
      )
ORDER BY CA_NM
```

This SQL-statement only selects user-defined conditional actions. In the catalog database these conditional actions coincide with the user-defined constraints that were discussed in the manifesto. Constraints of type E, R, D C, alternate keys and mark constraints (see section 7) are not shown here. Because of their fixed structure, the specifications of these constraints can be derived from the catalog data. To be specific, R-tables KEY and KEY_COLUMN specify entity integrity (E), referential integrity (R) and alternate key

constraints. R-tables DOMAIN and DOM_RANGE specify domain (D) constraints. Mark constraints are defined by the values of columns A_MARK_ALL and I_MARK_ALL in the R-table COLUMN.

Only the specifications of the few column (C) constraints, designated by column CA_NM_C in the R-table COLUMN, cannot be inferred from the contents of the catalog tables. In the case of the RM/V2 catalog database all column constraints apply to columns that belong to the domain RL_EXPR that contains correct relational expressions. The column constraints on this domain limit the expressions that are allowed for the column. For example, the column RL_EXPR in the relation R_TABLE cannot be allowed to contain an INSERT-statement. The meaning of the column constraints follows from the column description in the main text.

The contents of the COND_ACT table specifies the timing type (TT or TC) and the constraint type. Although the catalog database does not contain functional dependency, multi-valued dependency or join dependency constraints, it does constain some inclusion dependencies.

COND_ACT

CA_NM	TT	CONSTR_TYP
AR01	TT	
AR02	TT	
AR03	TT	
AU01	TT	
AU02	TT	
AU03	TT	
AU04	TT	
AU05	TT	
AU06	TT	
AU07	TT	
AU08	TT	
AU09	TT	
AU10	TT	
AU11	TT	
AU12	TT	
AU13	TT	
AU14	TT	
AU15	TT	
BM01	TT	INCL_DPCY
BM02	TC	
BM03	TT	
BM04	TT	
BM05	TT	
BM06	TT	
BM07	TT	INCL_DPCY
BM08	TT	
BM09	TT	
BM10	TT	
BM11	TT	
BM12	TT	
BM13	TT	
BM14	TT	

CA_NM	TT	CONSTR_TYPE
BM15	TT	
BM16	TT	
BM17	TT	
BM18	TT	
CA01	TT	
CA02	TT	
CA03	TT	
CA04	TT	
CA05	TT	
CA06	TT	
CA07	TT	
CA08	TT	
CA09	TT	
CA10	TT	
CA11	TT	
CA12	TT	
CA13	TT	
CA14	TT	
CA15	TT	
CA16	TT	
CA17	TT	
CA18	TT	
CA19	TT	
CA20	TT	
CA21	TT	
CA22	TT	
CA23	TT	
CA24	TT	
CA25	TT	
CA26	TT	
CA27	TT	
CA28	TT	

CA_NM	TT	CONSTR_TYPE
CM01	TT	
CM02	TT	
CM03	TT	
CM04	TT	
CM05	TT	
CM06	TT	
CM07	TT	
CM08	TT	
CM09	TT	
DD01	TT	INCL_DPCY
DD02	TT	INCL_DPCY
DD03	TT	
DD04	TT	
DD05	TT	INCL_DPCY
DD06	TT	INCL_DPCY
DD07	TT	
DD08	TT	
DD09	TT	
DD10	TT	
DD11	TT	
DD12	TT	INCL_DPCY
DD13	TT	
DD14	TT	
DD15	TT	
DD16	TT	
DD17	TT	
FU01	TT	
FU02	TT	
FU03	TT	
FU04	TT	
FU05	TT	
FU06	TT	

CA_NM	TT	CONSTR_TYPE
FU07	TT	
FU08	TT	
FU09	TT	
FU10	TT	
FU11	TT	
FU12	TT	
IX01	TT	
IX02	TT	
IX03	TT	
IX04	TT	
IX05	TT	
IX06	TT	
IX07	TT	
IX08	TT	
LG01	TT	
LG02	TT	INCL_DPCY
LG03	TT	
LG04	TT	
NR01	TT	
NR02	TT	
NR03	TT	
NR04	TT	
NR05	TT	
NR06	TT	
ST01	TT	
ST02	TT	
ST03	TT	
VW01	TT	
VW02	TT	
VW03	TT	
VW04	TT	
VW05	TT	

The data from the RT_CA relation show what DML-operations on an R-table that is subject to a conditional action trigger that conditional action. The constraints correspond to the user-defined constraints that were discussed in the manifesto.

The RT_CA data are retrieved by means of the following SQL statement:

```
SELECT CA_NM, RT_NM, INS, DEL
FROM RT_CA
WHERE USER_NM_RT = 'RMV2'    /* username of catalog owner */
  AND USER_NM_CA = 'RMV2'
ORDER BY CA_NM
```

RT_CA

CA_NM	RT_NM	INS	DEL
AR01	ARCHIVE	YES	NO
AR02	RT_SITE	YES	NO
AR03	KEY	YES	NO
AU01	RT_SITE	YES	NO
	USER_SITE	NO	YES
AU02	COLUMN	YES	NO
AU03	RT_AUTH	YES	NO
AU04	RT_AUTH	YES	NO
AU05	RT_AUTH	YES	NO
AU06	RT_AUTH	YES	NO
AU07	COL_AUTH	YES	NO
AU08	COL_AUTH	YES	NO
AU09	TERM_DFLT	YES	NO
AU10	PROG_DFLT	YES	NO
AU11	USER	YES	NO
AU12	USER_SITE	YES	NO
AU13	KEY	NO	YES
	RT_AUTH	YES	NO
AU14	RT_AUTH	YES	NO
AU15	USER_SITE	YES	NO
BM01	COLUMN	NO	YES
	R_TABLE	YES	NO
BM02	R_TABLE	YES	NO
BM03	KEY	NO	YES
	R_TABLE	YES	NO
BM04	KEY	YES	NO
BM05	KEY	YES	NO
	KEY_REF	YES	NO
BM06	KEY	NO	YES
	KEY_REF	YES	NO
BM07	KEY	YES	NO
	KEY_COLUMN	NO	YES
BM08	COLUMN	NO	YES
	KEY_COLUMN	YES	YES
	KEY	YES	YES
	KEY_REF	YES	YES
BM09	KEY	NO	NO
	KEY_COLUMN	YES	YES
BM10	R_TABLE	YES	NO
BM11	COLUMN	YES	YES
BM12	KEY_COLUMN	YES	YES
BM13	COLUMN	NO	NO
	KEY_COLUMN	YES	NO
	KEY	NO	NO
BM14	COLUMN	NO	NO
	KEY_COLUMN	YES	NO
	KEY	NO	NO
BM15	DOM_RANGE	YES	NO
BM16	DOM_RANGE	YES	YES
BM17	DOM_RANGE	YES	NO
BM18	DOM_RANGE	YES	NO
CA01	COLUMN	YES	NO
	KEY	YES	NO
	DOMAIN	YES	NO
	COND_ACT	YES	NO
	COL_CA	YES	NO
	RT_CA	YES	NO
CA02	COLUMN	YES	NO
	COL_CA	YES	NO
	KEY	YES	NO
	DOMAIN	YES	NO
	RT_CA	YES	NO
CA03	COLUMN	YES	NO
	DOMAIN	YES	NO
	KEY	YES	NO
	RT_CA	YES	NO
	COL_CA	YES	NO
CA04	COLUMN	YES	NO
	RT_CA	YES	NO
	DOMAIN	YES	NO
	KEY	YES	NO
	COL_CA	YES	NO
....	.....	...	...

CA_NM	RT_NM	INS	DEL
CA05	COLUMN	YES	NO
	COL_CA	YES	NO
	RT_CA	YES	NO
	DOMAIN	YES	NO
	KEY	YES	NO
CA06	COLUMN	YES	NO
	COL_CA	YES	NO
	DOMAIN	YES	NO
	KEY	YES	NO
	RT_CA	YES	NO
CA07	COLUMN	YES	NO
CA08	COLUMN	YES	NO
	DOMAIN	YES	NO
CA09	COLUMN	YES	NO
	COL_CA	YES	NO
	RT_CA	YES	NO
	KEY	YES	NO
CA10	COND_ACT	YES	NO
CA11	COND_ACT	YES	NO
CA12	COND_ACT	YES	NO
CA13	COND_ACT	YES	NO
CA14	KEY	YES	NO
CA15	KEY_COLUMN	YES	NO
CA16	KEY	YES	NO
CA17	KEY	YES	NO
CA18	COND_ACT	YES	NO
CA19	COL_CA	YES	NO
CA20	COL_CA	YES	YES
	COND_ACT	YES	NO
	RT_CA	YES	YES
CA21	COL_CA	YES	YES
	COND_ACT	YES	NO
	RT_CA	YES	YES
CA22	COL_CA	YES	YES
	COND_ACT	YES	NO
	RT_CA	YES	YES
CA23	COL_CA	NO	YES
	KEY_COLUMN	YES	NO
CA24	COL_CA	YES	YES
CA25	COL_CA	YES	NO
CA26	COL_CA	YES	YES
	RT_CA	YES	YES
CA27	COL_CA	YES	YES
	KEY_COLUMN	YES	YES
CA28	DOMAIN	YES	NO
CM01	COLUMN_STRUCT	YES	NO
CM02	DOMAIN_STRUCT	YES	NO
CM03	COLUMN_STRUCT	YES	YES
CM04	DOMAIN_STRUCT	YES	YES
CM05	COLUMN	NO	YES
	DOMAIN_STRUCT	YES	YES
	COLUMN_STRUCT	YES	YES
CM06	COLUMN_STRUCT	YES	NO
CM07	DOMAIN	NO	YES
	DOMAIN_STRUCT	NO	YES
CM08	DOMAIN	YES	NO
	DOMAIN_STRUCT	YES	YES
CM09	COLUMN_STRUCT	YES	NO
	KEY_COLUMN	YES	NO
DD01	RT_SITE	NO	YES
	R_TABLE	YES	NO
DD02	COL_SITE	NO	YES
	RT_SITE	YES	NO
DD03	RT_SITE	YES	NO
DD04	SITE	NO	YES
DD05	USER	YES	NO
	USER_SITE	NO	YES
DD06	SITE	YES	NO
	USER_SITE	NO	YES
DD07	USER	YES	NO
	USER_SITE	NO	YES
DD08	SINGULAR	YES	NO
....	.....	...	...

CA_NM	RT_NM	INS	DEL
DD09	USER_SITE	YES	NO
DD10	FUNCTION	YES	NO
	FUNC_SITE	NO	YES
DD11	FUNC_SITE	NO	YES
	RT_FUNC	YES	NO
DD12	CA_SITE	NO	YES
	COND_ACT	YES	NO
DD13	CA_SITE	YES	NO
	KEY_COLUMN	YES	NO
	COL_SITE	NO	YES
DD14	RT_CA	YES	YES
	RT_SITE	YES	YES
DD15	COL_CA	YES	YES
	COL_SITE	YES	YES
DD16	COLUMN	YES	YES
	COL_SITE	YES	YES
DD17	CA_SITE	YES	NO
	DOMAIN	YES	NO
	SITE	YES	NO
FU01	FUNCTION	YES	YES
FU02	FUNCTION	YES	YES
FU03	FUNCTION	YES	NO
FU04	FUNCTION	YES	YES
FU05	R_TABLE	YES	NO
FU06	RT_FUNC	NO	YES
	R_TABLE	YES	NO
FU07	ARGUMENT	YES	YES
	FUNCTION	YES	YES
FU08	ARGUMENT	YES	YES
FU09	SINGULAR	YES	YES
FU10	SINGULAR	YES	NO
FU11	FUNCTION	YES	NO
FU12	DOMAIN	YES	NO
IX01	DB_INDEX	YES	NO
	KEY	YES	NO
IX02	DB_INDEX	YES	NO
	DB_INDEX_COL	NO	YES
IX03	DB_INDEX_COL	YES	NO
IX04	DB_INDEX	YES	NO
	DB_INDEX_COL	YES	YES
IX05	DB_INDEX_COL	YES	NO
IX06	KEY	YES	NO
IX07	KEY	YES	NO
IX08	KEY_COLUMN	YES	YES
LG01	AUDIT_LOG	YES	NO
LG02	AUDIT_LOG	YES	NO
	LOG_PK_COLUMN	NO	YES
LG03	AUDIT_LOG	YES	NO
	LOG_PK_COLUMN	YES	NO
LG04	AUDIT_INTERVAL	YES	NO
NR01	RT_SYN	YES	NO
NR02	COLUMN_STRUCT	YES	NO
	RL_KEYWORD	YES	NO
NR03	FUNCTION	YES	NO
	R_TABLE	YES	NO
	RT_SYN	YES	NO
NR04	BASIC_DATATYPE	YES	NO
	R_TABLE	YES	NO
	RT_SYN	YES	NO
	COLUMN	YES	NO
	DOMAIN	YES	NO
	FUNCTION	YES	NO
NR05	COLUMN	YES	NO
	DOMAIN	YES	NO
NR06	COLUMN	YES	NO
	DOMAIN	YES	NO
ST01	RT_SITE	YES	NO
ST02	COL_SITE	YES	NO
ST03	COL_SITE	YES	NO
	RT_SITE	YES	NO
VW01	COLUMN	YES	NO
	R_TABLE	YES	NO
....	.....	...	...

RT_CA (continued)

VW02	COLUMN	YES	NO
VW03	COLUMN	YES	NO
VW04	COLUMN	YES	NO
	DOMAIN	NO	YES
VW05	COLUMN	YES	YES

Finally, the COL_CA data are retrieved by means of the following SQL statement:

```
SELECT CA_NM, RT_NM, COL_NM, GROUP_ID
FROM COL_CA
WHERE USER_NM_RT = 'RMV2'    /* username of catalog owner */
  AND USER_NM_CA = 'RMV2'
ORDER BY CA_NM, RT_NM
```

COL_CA

CA_NM	RT_NM	COL_NM	GROUP_ID
AR01	ARCHIVE	CR_DATE	
		CR_TIME	
AR02	RT_SITE	CR_DATE	
		CR_TIME	
AR03	KEY	KEY_TYPE	
		USER_NM	
		RT_NM	
	R_TABLE	RT_NM	
		RT_TYPE	
		USER_NM	
AU01	RT_SITE	N_DELAY	
		N_DELETE	
		SITE_NM	
		N_DROP	
	USER	USER_GRP_NM	
		USER_NM	
	USER_SITE	SITE_NM	
		USER_NM	
AU02	COLUMN	A_MARK_ALL	
		MARK_PREF	
		I_MARK_ALL	
AU03	RT_AUTH	USER_NM_GRANTEE	
		USER_NM_GRANTOR	
		USER_NM_RT	
AU04	RT_AUTH	RT_NM	
		USER_NM_GRANTEE	
		USER_NM_GRANTOR	
		USER_NM_RT	
AU05	RT_AUTH	ARCH_ALL	
		DEL_ALL	
		PK_UPD_ALL	
		RL_EXPR_DEL	
		RL_EXPR_INS	
		RL_EXPR_PK_UPD	
		RL_EXPR_RET	
		RL_EXPR_SNAPSHOT	
		SNAPSHOT_ALL	
		RL_EXPR_CHECKOUT	
		RL_EXPR_ARCH	
		RET_ALL	
		INS_ALL	
		CHECKOUT_ALL	
AU06	RT_AUTH	RET_ALL	
		SNAPSHOT_ALL	
AU07	COL_AUTH	RL_EXPR_UPD	
		UPD_ALL	
AU08	COLUMN	COL_NM	
		RT_NM	
		DOM_NM	
		USER_NM	
	COL_AUTH	COL_NM	
		USER_NM_RT	
		DEFAULT_VAL	
		RT_NM	
....	.....	.....	.....

CA_NM	RT_NM	COL_NM	GROUP_ID
	DOM_RANGE	DOM_NM	
		HIVAL	
		LOVAL	
AU09	COLUMN	COL_NM	
		RT_NM	
		DOM_NM	
		USER_NM	
	DOM_RANGE	DOM_NM	
		HIVAL	
		LOVAL	
	TERM_DFLT	COL_NM	
		DEFAULT_VAL	
		RT_NM	
		USER_NM	
AU10	COLUMN	COL_NM	
		RT_NM	
		DOM_NM	
		USER_NM	
	DOM_RANGE	DOM_NM	
		LOVAL	
		HIVAL	
	PROG_DFLT	COL_NM	
		DEFAULT_VAL	
		RT_NM	
		USER_NM	
AU11	USER	USER_GRP_NM	
		USER_NM	
AU12	USER_SITE	MAX_STOR_QUERY_ACT	
		MAX_STOR_TOTAL_ACT	
		MAX_STOR_TOTAL_ANT	
		MAX_STOR_QUERY_ANT	
AU13	KEY	KEY_TYPE	
		RT_NM	
		USER_NM	
	RT_AUTH	PK_HIDDEN	
		USER_NM_RT	
		RT_NM	
AU14	RT_AUTH	CHECKOUT_ALL	
		USER_NM_RT	
		RT_NM	
	R_TABLE	RT_NM	
		USER_NM	
		RT_TYPE	
AU15	USER_SITE	GRANT_ALL	
		IND_DBA_USER	
BM01	COLUMN	RT_NM	SUPER
		USER_NM	SUPER
	R_TABLE	RT_NM	SUB
		USER_NM	SUB
BM02	R_TABLE	RL_EXPR	
BM03	KEY	KEY_TYPE	
		USER_NM	
		RT_NM	
	R_TABLE	RT_TYPE	
....	.....	.....	.....

COL_CA (continued)

CA_NM	RT_NM	COL_NM	GROUP_ID
BM04	KEY	KEY_TYPE	
		RT_NM	
		USER_NM	
BM05	KEY	KEY_TYPE	
	KEY_REF	KEY_NM	
		RT_NM_FK	
		USER_NM_FK	
BM06	KEY	RT_NM	
		USER_NM	
	KEY_REF	RT_NM_PK	
		USER_NM_PK	
BM07	KEY	KEY_NM	SUB
		RT_NM	SUB
		USER_NM	SUB
	KEY_COLUMN	KEY_NM	SUPER
		RT_NM	SUPER
		USER_NM	SUPER
BM08	KEY	KEY_NM	
		RT_NM	
		USER_NM	
		KEY_TYPE	
	KEY_COLUMN	COL_NM	
		RT_NM	
		KEY_COL_SEQ#	
		USER_NM	
		KEY_NM	
	KEY_REF	KEY_NM	
		RT_NM_FK	
		RT_NM_PK	
		USER_NM_FK	
		USER_NM_PK	
BM09	KEY	KEY_TYPE	
	KEY_COLUMN	COL_NM	
		USER_NM	
		RT_NM	
		KEY_NM	
BM10	R_TABLE	RL_EXPR	
		RT_TYPE	
BM11	COLUMN	A_MARK_ALL	
		USER_NM	
		RT_NM	
		COL_SEQ#	
		I_MARK_ALL	
BM12	KEY_COLUMN	KEY_COL_SEQ#	
		USER_NM	
		KEY_NM	
		RT_NM	
BM13	COLUMN	COL_NM	
		RT_NM	
		USER_NM	
	KEY	KEY_TYPE	
		RT_NM	
		USER_NM	
	KEY_COLUMN	COL_NM	
		RT_NM	
		USER_NM	
		KEY_NM	
BM14	COLUMN	COL_NM	
		RT_NM	
		USER_NM	
	KEY	KEY_TYPE	
		RT_NM	
		USER_NM	
	KEY_COLUMN	COL_NM	
		RT_NM	
		USER_NM	
		KEY_NM	
BM15	DOM_RANGE	HIVAL	
		LOVAL	
BM16	DOM_RANGE	DOM_NM	
		DR_SEQ#	
BM17	DOM_RANGE	DOM_NM	
		LOVAL	
		DR_SEQ#	
		HIVAL	
....	.....	.....	.....

CA_NM	RT_NM	COL_NM	GROUP_ID
BM18	DOM_RANGE	HIVAL	
		LOVAL	
CA01	COLUMN	CA_NM_C	
		USER_NM	
		CA_NM_M	
	COL_CA	CA_NM	
		USER_NM_CA	
	COND_ACT	CA_NM	
		USER_NM	
		CONDITION	
	DOMAIN	CA_NM	
		USER_NM_CA	
	KEY	CA_NM	
		USER_NM	
	RT_CA	CA_NM	
		USER_NM_CA	
CA02	COLUMN	CA_NM_C	
		USER_NM	
		CA_NM_M	
	COL_CA	CA_NM	
		USER_NM_CA	
	COND_ACT	CA_NM	
		USER_NM	
	DOMAIN	CA_NM	
		USER_NM_CA	
	KEY	CA_NM	
		USER_NM	
	RT_CA	CA_NM	
		USER_NM_CA	
CA03	COLUMN	CA_NM_C	
		CA_NM_M	
		USER_NM	
	COL_CA	CA_NM	
		USER_NM_CA	
	COND_ACT	CA_NM	
		USER_NM	
	DOMAIN	CA_NM	
		USER_NM_CA	
	KEY	CA_NM	
		USER_NM	
	RT_CA	CA_NM	
		USER_NM_CA	
CA04	COLUMN	CA_NM_C	
		CA_NM_M	
		USER_NM	
	COL_CA	CA_NM	
		USER_NM_CA	
	COND_ACT	CA_NM	
		USER_NM	
	DOMAIN	CA_NM	
		USER_NM_CA	
	KEY	CA_NM	
		USER_NM	
	RT_CA	CA_NM	
		USER_NM_CA	
CA05	COLUMN	CA_NM_C	
		USER_NM	
		CA_NM_M	
	COL_CA	CA_NM	
		USER_NM_CA	
	COND_ACT	CA_NM	
		USER_NM	
	DOMAIN	CA_NM	
		USER_NM_CA	
	KEY	CA_NM	
		USER_NM	
	RT_CA	CA_NM	
		USER_NM_CA	
CA06	COLUMN	CA_NM_C	
		USER_NM	
		CA_NM_M	
	COL_CA	CA_NM	
		USER_NM_CA	
	COND_ACT	CA_NM	
		USER_NM	
....	.....	.....	.....

COL_CA (continued)

CA_NM	RT_NM	COL_NM	GROUP_ID
	DOMAIN	CA_NM	
		USER_NM_CA	
	KEY	CA_NM	
		USER_NM	
	RT_CA	CA_NM	
		USER_NM_CA	
CA07	COLUMN	A_MARK_ALL	
		I_MARK_ALL	
		CA_NM_M	
CA08	COLUMN	CA_NM_C	
		USER_NM	
	COND_ACT	CA_NM	
		USER_NM	
		TTIME	
	DOMAIN	CA_NM	
		USER_NM_CA	
CA09	COLUMN	CA_NM_C	
		USER_NM	
		RT_NM	
		CA_NM_M	
	COL_CA	CA_NM	
		USER_NM_CA	
		USER_NM_RT	
		RT_NM	
		COL_NM	
	COND_ACT	CA_NM	
		USER_NM	
		CONDITION	
	KEY	CA_NM	
		KEY_TYPE	
		KEY_NM	
		RT_NM	
		USER_NM	
	RT_CA	CA_NM	
		USER_NM_CA	
		USER_NM_RT	
		RT_NM	
	R_TABLE	RT_NM	
		USER_NM	
		RT_TYPE	
CA10	COND_ACT	CA_DATE	
		CA_TIME	
		INTERVAL	
CA11	COND_ACT	CA_DATE	
		CONDITION	
		CONSTR_TYPE	
		IND_STATIC	
CA12	COND_ACT	CA_DATE	
		CA_TIME	
CA13	COND_ACT	CONDITION	
		TTIME	
CA14	KEY	CA_NM	
		KEY_TYPE	
CA15	COLUMN	CA_NM_M	
		USER_NM	
		RT_NM	
		COL_NM	
	KEY	KEY_NM	
		RT_NM	
		USER_NM	
		KEY_TYPE	
	KEY_COLUMN	COL_NM	
		RT_NM	
		KEY_NM	
		USER_NM	
CA16	KEY	DEL_ACT	
		UPD_ACT	
		KEY_TYPE	
CA17	COND_ACT	CA_NM	
		USER_NM	
		TTIME	
	KEY	CA_NM	
		USER_NM	
		KEY_TYPE	
....	.....	.....	.....

CA_NM	RT_NM	COL_NM	GROUP_ID
CA18	COND_ACT	CONDITION	
		CONSTR_TYPE	
CA19	COL_CA	CA_NM	
		USER_NM_CA	
		GROUP_ID	
	COND_ACT	CA_NM	
		USER_NM	
		CONSTR_TYPE	
CA20	COLUMN	CA_NM_C	
		CA_NM_M	
		USER_NM	
	COL_CA	CA_NM	
		USER_NM_CA	
		GROUP_ID	
	COND_ACT	CA_NM	
		USER_NM	
		CONSTR_TYPE	
	DOMAIN	CA_NM	
		USER_NM_CA	
	KEY	CA_NM	
		USER_NM	
	RT_CA	DEL	
		INS	
		USER_NM_CA	
		RET	
		RT_NM	
CA21	COLUMN	CA_NM_C	
		USER_NM	
		CA_NM_M	
	COL_CA	CA_NM	
		GROUP_ID	
		USER_NM_CA	
	COND_ACT	CA_NM	
		USER_NM	
		CONSTR_TYPE	
	DOMAIN	CA_NM	
		USER_NM_CA	
	KEY	CA_NM	
		USER_NM	
	RT_CA	DEL	
		INS	
		RT_NM	
		USER_NM_CA	
		RET	
CA22	COLUMN	CA_NM_C	
		USER_NM	
		CA_NM_M	
	COL_CA	CA_NM	
		GROUP_ID	
		USER_NM_CA	
	COND_ACT	CA_NM	
		CONSTR_TYPE	
		USER_NM	
	DOMAIN	CA_NM	
		USER_NM_CA	
	KEY	CA_NM	
		USER_NM	
	RT_CA	DEL	
		RET	
		INS	
		RT_NM	
		USER_NM_CA	
CA23	COL_CA	CA_NM	
		GROUP_ID	
		COL_NM	
		USER_NM_RT	
		RT_NM	
		USER_NM_CA	
	COND_ACT	CA_NM	
		CONSTR_TYPE	
		USER_NM	
	KEY	KEY_NM	
		RT_NM	
		KEY_TYPE	
....	.....	.....	.....

COL_CA (continued)

CA_NM	RT_NM	COL_NM	GROUP_ID
		USER_NM	
	KEY_COLUMN	COL_NM	
		RT_NM	
		USER_NM	
		KEY_NM	
CA24	COLUMN	COL_NM	
		USER_NM	
		DOM_NM	
		RT_NM	
	COL_CA	COL_NM	
		RT_NM	
		USER_NM_RT	
		GROUP_ID	
		USER_NM_CA	
	COND_ACT	CA_NM	
		CONSTR_TYPE	
		USER_NM	
CA25	COL_CA	CA_NM	
		USER_NM_CA	
		GROUP_ID	
	COND_ACT	CA_NM	
		USER_NM	
		CONSTR_TYPE	
CA26	COL_CA	CA_NM	
		USER_NM_CA	
		GROUP_ID	
	COND_ACT	CA_NM	
		CONSTR_TYPE	
		USER_NM	
	RT_CA	CA_NM	
		RET	
		USER_NM_CA	
		INS	
		DEL	
CA27	COL_CA	CA_NM	
		COL_NM	
		GROUP_ID	
		USER_NM_CA	
		RT_NM	
		USER_NM_RT	
	COND_ACT	CA_NM	
		USER_NM	
		CONSTR_TYPE	
	KEY	KEY_NM	
		RT_NM	
		KEY_TYPE	
		USER_NM	
	KEY_COLUMN	COL_NM	
		USER_NM	
		RT_NM	
		KEY_NM	
	KEY_REF	KEY_NM	
		RT_NM_FK	
		RT_NM_PK	
		USER_NM_FK	
		USER_NM_PK	
CA28	DOMAIN	CA_NM	
		USER_NM_CA	
CM01	COLUMN_STRUCT	COL_NM_SUB	
		RT_NM	
		COL_NM_SUP	
		USER_NM	
CM02	DOMAIN_STRUCT	DOM_NM_SUB	
		DOM_NM_SUP	
CM03	COLUMN_STRUCT	COL_NM_SUP	
		RT_NM	
		CS_SEQ#	
		USER_NM	
CM04	DOMAIN_STRUCT	DOM_NM_SUP	
		DS_SEQ#	
CM05	COLUMN	COL_NM	
		RT_NM	
		USER_NM	
		DOM_NM	

CA_NM	RT_NM	COL_NM	GROUP_ID
	COLUMN_STRUCT	COL_NM_SUB	
		USER_NM	
		CS_SEQ#	
		RT_NM	
		COL_NM_SUP	
	DOMAIN_STRUCT	DOM_NM_SUB	
		DOM_NM_SUP	
		DS_SEQ#	
CM06	COLUMN	A_MARK_ALL	
		COL_NM	
		USER_NM	
		RT_NM	
		I_MARK_ALL	
	COLUMN_STRUCT	COL_NM_SUB	
		COL_NM_SUP	
		USER_NM	
		RT_NM	
CM07	DOMAIN	COMP_ALL	
		DOM_NM	
	DOMAIN_STRUCT	DOM_NM_SUB	
		DOM_NM_SUP	
CM08	DOMAIN	BD_TYPE_NM	
		LENGTH	
		SCALE	
		DOM_NM	
	DOMAIN_STRUCT	DOM_NM_SUP	
CM09	COLUMN	COL_NM	
		USER_NM	
		RT_NM	
	COLUMN_STRUCT	COL_NM_SUP	
		USER_NM	
		RT_NM	
	KEY_COLUMN	COL_NM	
		RT_NM	
		USER_NM	
DD01	RT_SITE	RT_NM	SUPER
		USER_NM	SUPER
	R_TABLE	RT_NM	SUB
		USER_NM	SUB
DD02	COL_SITE	RT_NM	SUPER
		SITE_NM	SUPER
		USER_NM	SUPER
	RT_SITE	RT_NM	SUB
		USER_NM	SUB
		SITE_NM	SUB
DD03	RT_SITE	IND_REPLICA	
		RL_EXPR_LG	
		RL_EXPR_GL	
		USER_NM	
		RT_NM	
	R_TABLE	RT_NM	
		RT_TYPE	
		USER_NM	
DD04	SITE	IND_CATALOG_SITE	
DD05	USER	USER_NM	SUB
	USER_SITE	USER_NM	SUPER
DD06	SITE	SITE_NM	SUB
	USER_SITE	SITE_NM	SUPER
DD07	SITE	SITE_NM	
	USER	IND_GDBA	
		USER_GRP_NM	
		USER_NM	
	USER_SITE	SITE_NM	
		USER_NM	
DD08	SINGULAR	USER_NM_CATALOG	
	USER	IND_GDBA	
		USER_NM	
DD09	USER	IND_GDBA	
		USER_NM	
	USER_SITE	IND_DBA_USER	
		USER_NM	
DD10	FUNCTION	FUNC_NM	
		USER_NM	
	FUNC_SITE	FUNC_NM	

COL_CA (continued)

CA_NM	RT_NM	COL_NM	GROUP_ID
		USER_NM	
		IND_STORED	
DD11	FUNCTION	FUNC_NM	
		USER_NM	
	FUNC_SITE	FUNC_NM	
		SITE_NM	
		USER_NM	
	RT_FUNC	FUNC_NM	
		USER_NM_FUNC	
		RT_NM	
	SITE	SITE_NM	
DD12	CA_SITE	CA_NM	SUPER
		USER_NM	SUPER
	COND_ACT	CA_NM	SUB
		USER_NM	SUB
DD13	CA_SITE	CA_NM	
		SITE_NM	
		USER_NM	
	COL_SITE	COL_NM	
		RT_NM	
		SITE_NM	
		USER_NM	
	KEY	KEY_NM	
		USER_NM	
		RT_NM	
	KEY_COLUMN	COL_NM	
		RT_NM	
		USER_NM	
		KEY_NM	
DD14	CA_SITE	CA_NM	
		USER_NM	
		SITE_NM	
	RT_CA	CA_NM	
		USER_NM_CA	
		USER_NM_RT	
		RT_NM	
	RT_SITE	RT_NM	
		SITE_NM	
		USER_NM	
DD15	CA_SITE	CA_NM	
		USER_NM	
		SITE_NM	
	COL_CA	CA_NM	
		COL_NM	
		RT_NM	
		USER_NM_RT	
		USER_NM_CA	
	COL_SITE	COL_NM	
		RT_NM	
		SITE_NM	
		USER_NM	
DD16	CA_SITE	CA_NM	
		SITE_NM	
		USER_NM	
	COLUMN	CA_NM_C	
		RT_NM	
		CA_NM_M	
		USER_NM	
		COL_NM	
	COL_CA	CA_NM	
		USER_NM_CA	
		COL_NM	
		USER_NM_RT	
		RT_NM	
DD17	CA_SITE	CA_NM	
		USER_NM	
	DOMAIN	CA_NM	
		USER_NM_CA	
FU01	FUNCTION	FUNC_NM	
		USER_NM	
		FUNC_NM_INV	
FU02	FUNCTION	FUNC_NM	
		USER_NM	
		FUNC_NM_INV	
....	.....	.....	.....

CA_NM	RT_NM	COL_NM	GROUP_ID
FU03	FUNCTION	FUNC_NM_DOD	
		IND_DOD	
FU04	FUNCTION	FUNC_NM	
		FUNC_NM_DOD	
		USER_NM	
		IND_DOD	
FU05	R_TABLE	FUNC_NM_VW	
		RT_TYPE	
		INS_ALL	
FU06	RT_FUNC	FUNC_NM	
		USER_NM_FUNC	
		USER_NM_RT	
		RT_NM	
	R_TABLE	FUNC_NM_VW	
		USER_NM	
		RT_NM	
FU07	ARGUMENT	ARG_NM	
		FUNC_NM	
		USER_NM	
		ARG_SEQ#	
	FUNCTION	FUNC_NM	
		FUNC_NM_INV	
		FUNC_NM_DOD	
		USER_NM	
FU08	ARGUMENT	ARG_SEQ#	
		USER_NM	
		FUNC_NM	
FU10	SINGULAR	DOM_NM_ARG	
		DOM_NM_FUNC	
FU11	FUNCTION	HOST_LANG	
		ROUT_EXEC	
		ROUT_SRC	
		USER_NM	
	SINGULAR	USER_NM_CATALOG	
FU12	DOMAIN	DOM_NM	
		USER_NM_CA	
		USER_NM_DOM	
	SINGULAR	DOM_NM_ARG	
		DOM_NM_FUNC	
		USER_NM_CATALOG	
IX01	DB_INDEX	INDEX_NM	
		USER_NM	
	KEY	INDEX_NM	
		USER_NM	
IX02	DB_INDEX	INDEX_NM	
		USER_NM	
	DB_INDEX_COL	INDEX_NM	
		USER_NM_DBI	
IX03	COLUMN	COL_NM	
		RT_NM	
		DOM_NM	
		USER_NM	
	DB_INDEX	DOM_NM	
		USER_NM	
		INDEX_NM	
	DB_INDEX_COL	COL_NM	
		RT_NM	
		USER_NM_RT	
		INDEX_NM	
		USER_NM_DBI	
IX04	DB_INDEX	INDEX_NM	
		USER_NM	
	DB_INDEX_COL	COL_NM	
		USER_NM_DBI	
		USER_NM_RT	
		INDEX_NM	
		RT_NM	
IX05	DB_INDEX_COL	RT_NM	
		USER_NM_RT	
	R_TABLE	RT_NM	
		RT_TYPE	
		USER_NM	
IX06	KEY	INDEX_NM	
		RT_NM	
....	.....	.....	.....

COL_CA (continued)

CA_NM	RT_NM	COL_NM	GROUP_ID
		USER_NM	
	R_TABLE	RT_NM	
		RT_TYPE	
		USER_NM	
IX07	KEY	INDEX_NM	
		KEY_TYPE	
IX08	KEY	INDEX_NM	
		RT_NM	
		USER_NM	
		KEY_NM	
	KEY_COLUMN	COL_NM	
		RT_NM	
		USER_NM	
		KEY_NM	
		KEY_COL_SEQ#	
LG01	AUDIT_LOG	RT_NM	
		USER_NM_RT	
	R_TABLE	RT_NM	
		RT_TYPE	
		USER_NM	
LG02	AUDIT_LOG	AUDIT_SEQ#	SUB
	LOG_PK_COLUMN	AUDIT_SEQ#	SUPER
LG03	AUDIT_LOG	RT_NM	
		USER_NM_RT	
	LOG_PK_COLUMN	RT_NM	
		USER_NM	
LG04	AUDIT_INTERVAL	DATE_LOG_OFF	
		USER_NM_LOG_OFF	
		TIME_LOG_OFF	
NR01	RT_SYN	RT_NM	
		USER_NM_SYN	
		USER_NM_RT	
		RT_NM_SYN	
NR02	COLUMN_STRUCT	COL_NM_SUP	
	RL_KEYWORD	RL_KEYW	
NR03	DOMAIN	DOM_NM	
	FUNCTION	FUNC_NM	
	RT_SYN	RT_NM_SYN	
	R_TABLE	RT_NM	
NR04	BASIC_DATATYPE	BD_TYPE_NM	
	COLUMN	COL_NM	
		USER_NM	
	DOMAIN	DOM_NM	
	RT_SYN	RT_NM_SYN	
		USER_NM_SYN	
	R_TABLE	RT_NM	
		USER_NM	
NR05	BASIC_DATATYPE	BD_TYPE_NM	
		IND_LENGTH	
	COLUMN	BD_TYPE_NM	
		LENGTH	
	DOMAIN	BD_TYPE_NM	
		LENGTH	
NR06	BASIC_DATATYPE	BD_TYPE_NM	
		IND_SCALE	
	COLUMN	BD_TYPE_NM	
		SCALE	
	DOMAIN	BD_TYPE_NM	
		SCALE	
ST01	RT_SITE	IND_REPLICA	
		RL_EXPR_LG	
		ROWCOUNT	
ST02	COL_SITE	DCOLCOUNT	
		RT_NM	
		USER_NM	
		SITE_NM	
	RT_SITE	ROWCOUNT	
		USER_NM	
		SITE_NM	
		RT_NM	
ST03	COL_SITE	DCOLCOUNT	
		USER_NM	
		RT_NM	
		SITE_NM	
	RT_SITE	ROWCOUNT	
		SITE_NM	
		USER_NM	
		RT_NM	
VW01	COLUMN	RT_NM	
		UPD_ALL	
		USER_NM	
	R_TABLE	DEL_ALL	
		RT_NM	
		RT_TYPE	
		USER_NM	
		INS_ALL	
VW02	COLUMN	COL_NM_BASE	
		RT_NM_BASE	
		USER_NM_BASE	
VW03	COLUMN	RT_NM	
		USER_NM	
		RT_NM_BASE	
	R_TABLE	RT_NM	
		USER_NM	
		RT_TYPE	
VW04	COLUMN	DOM_NM	
		USER_NM	
		RT_NM	
	R_TABLE	RT_NM	
		RT_TYPE	
		USER_NM	
VW05	COLUMN	BD_TYPE_NM	
		LENGTH	
		SCALE	
		DOM_NM	
	DOMAIN	BD_TYPE_NM	
		SCALE	
		LENGTH	
		DOM_NM	

Appendix B User-defined Constraints in the Catalog Model

This appendix contains SQL-queries (wherever possible) for all user-defined constraints that were discussed in the manifesto. Although one may criticize the SQL-language on many grounds, formulating the constraints in SQL instead of Relational Algebra or Relational Calculus has been preferred, because SQL is easily interpreted by many people. Moreover, using SQL has facilitated the implementation and validation of the constraints. The queries have been implemented and tested using an ORACLE™ RDBMS. Because the R-table names *COLUMN* and *USER* are ORACLE keywords, other names have been used. However, in the following SQL-statements the original names have been reused, in order to retain a match with the manifesto. In addition, two ORACLE-specific features have been used, namely the system-defined dummy table DUAL that contains one tuple and a system-defined variable containing the current date.

Perhaps more interesting is the observation that only an RM/V2-version of the SQL-language is able to express many of the user-defined constraints. To be specific, current SQL does not distinguish between A-marks and I-marks and is unable to perform domain checks, let alone domain check overrides. Wherever these problems are encountered, an appropriate comment is added to the SQL statement.

Because SQL does not explicitly support the universal quantifier (FOR ALL), the constraints have not been coded as assertions, but as their opposites. Thus, if a statement returns no tuples, the constraint it expresses is not violated. If the constraint is violated, an error message is returned.

In many cases the scope of a constraint is quite arbitrary. This means that many constraints can be split up or combined with other constraints. Although implementation of an RM/V2 catalog may call for an elementary approach, we have preferred to choose constraint scopes that limit the total number of constraints and provide relatively easy interpretation of the equivalent formulations in english.

The performance of the SQL statements in the ORACLE RDBMS environment is sometimes quite poor. Again, reader performance (readability) has been preferred over system performance. Besides, such low performance is an omission of ORACLE's optimizer, not of the programmer.

This appendix describes only the user-defined constraints with respect to the catalog model. Constraints of types E, R, C, D, and mark-constraints are not listed here. Their generalized nature would lead to a large number of very similar statements. Moreover, these constraints, together with a number of alternate key constraints are to be found in appendix A.

2. THE BASIC MODEL

BM01- Every tuple in R_TABLE is referenced by at least one COLUMN-tuple.

```
SELECT DISTINCT 'BM01: R_TABLE-TUPLE NOT REFERENCED BY COLUMN-TUPLE' FROM R_TABLE
WHERE (USER_NM, RT_NM) NOT IN
        (SELECT USER_NM, RT_NM FROM COLUMN)
```

BM02- No view definition (RL_EXPR) is directly or indirectly recursive.
This constraint cannot be expressed in RL.

BM03- Every tuple in R_TABLE for which RT_TYPE = 'BASE' or 'CHECKOUT' (see section 4) is referenced by a KEY-tuple having KEY_TYPE = 'PRIMARY'.

```
SELECT DISTINCT 'BM03: BASE_TABLE OR CHECKOUT DOES NOT HAVE A PRIMARY KEY'
FROM R_TABLE
WHERE RT_TYPE IN ('BASE', 'CHECKOUT')
  AND (USER_NM, RT_NM) NOT IN
        (SELECT USER_NM, RT_NM FROM KEY WHERE KEY_TYPE = 'PRIMARY')
```

BM04- The subset of KEY-tuples for which KEY_TYPE = 'PRIMARY' does not contain duplicate values for USER_NM and RT_NM.

```
SELECT DISTINCT 'BM04: R_TABLE HAS MORE THAN ONE PRIMARY KEY'
FROM KEY
WHERE KEY_TYPE = 'PRIMARY'
GROUP BY USER_NM, RT_NM
HAVING COUNT(*) > 1
```

BM05- Every tuple in KEY having KEY_TYPE = 'FOREIGN' is referenced by at least one KEY_REF-tuple, while other tuples in KEY are not referenced by KEY_REF-tuples.

```
SELECT DISTINCT 'BM05: FOREIGN KEY HAS NO REFERENCES OR NON FOREIGN KEY HAS'
FROM KEY
WHERE
   (    KEY_TYPE = 'FOREIGN'
    AND (USER_NM, RT_NM, KEY_NM) NOT IN
          (SELECT USER_NM_FK, RT_NM_FK, KEY_NM FROM KEY_REF)
   )
   OR
   (    KEY_TYPE <> 'FOREIGN'
    AND (USER_NM, RT_NM, KEY_NM) IN
          (SELECT USER_NM_FK, RT_NM_FK, KEY_NM FROM KEY_REF)
   )
```

BM06- Every tuple in KEY_REF references a tuple in R_TABLE that is referenced by a tuple in KEY having KEY_TYPE = 'PRIMARY'.

```
SELECT DISTINCT 'BM06: REFERENCED R_TABLE DOES NOT HAVE A PRIMARY KEY'
FROM KEY_REF
WHERE (USER_NM_PK, RT_NM_PK) NOT IN
       (SELECT USER_NM, RT_NM
        FROM KEY
        WHERE KEY_TYPE = 'PRIMARY'
       )
```

BM07- Every tuple in KEY is referenced by at least one KEY_COLUMN-tuple.

```
SELECT DISTINCT 'BM07: KEY-TUPLE IS NOT REFERENCED BY KEY_COLUMN-TUPLE'
FROM KEY
WHERE (USER_NM, RT_NM, KEY_NM) NOT IN
        (SELECT USER_NM, RT_NM, KEY_NM FROM KEY_COLUMN)
```

BM08- KEY-tuples with KEY_TYPE = 'PRIMARY', that are referenced via KEY_REF by a KEY-tuple with KEY_TYPE = 'FOREIGN', are referenced by pairs of KEY_COLUMN-tuples with corresponding values for KEY_COL_SEQ#, referencing COLUMN-tuples with the same value for DOM_NM.

```
SELECT DISTINCT 'BM08: UNMATCHED PRIMARY KEY FOREIGN KEY COMBINATIONS EXIST'
FROM KEY     FK,
     KEY_REF KR,
     KEY     PK
WHERE FK.KEY_TYPE    = 'FOREIGN'     /* foreign key identification */
  AND FK.USER_NM     = KR.USER_NM_FK /* join foreign key with key reference */
  AND FK.RT_NM       = KR.RT_NM_FK
  AND FK.KEY_NM      = KR.KEY_NM
  AND KR.USER_NM_PK  = PK.USER_NM    /* join key reference with primary key */
  AND KR.RT_NM_PK    = PK.RT_NM
  AND PK.KEY_TYPE    = 'PRIMARY'     /* primary key identification */
  AND (
        EXISTS
         (SELECT *
          FROM KEY_COLUMN FKC,
               COLUMN     FCOL
          WHERE FKC.USER_NM = FK.USER_NM
            AND FKC.RT_NM   = FK.RT_NM
            AND FKC.KEY_NM  = FK.KEY_NM
            AND FKC.USER_NM = FCOL.USER_NM
            AND FKC.RT_NM   = FCOL.RT_NM
            AND FKC.COL_NM  = FCOL.COL_NM
            AND NOT EXISTS
              (SELECT *
               FROM KEY_COLUMN PKC,
                    COLUMN     PCOL
               WHERE PKC.USER_NM       = PK.USER_NM
                 AND PKC.RT_NM         = PK.RT_NM
                 AND PKC.KEY_NM        = PK.KEY_NM
                 AND PKC.USER_NM       = PCOL.USER_NM
                 AND PKC.RT_NM         = PCOL.RT_NM
                 AND PKC.COL_NM        = PCOL.COL_NM
                 AND PKC.KEY_COL_SEQ#  = FKC.KEY_COL_SEQ#
                 AND FCOL.DOM_NM       = PCOL.DOM_NM))
      OR
        EXISTS
         (SELECT *
          FROM KEY_COLUMN PKC,
               COLUMN     PCOL
          WHERE PKC.USER_NM = PK.USER_NM
            AND PKC.RT_NM   = PK.RT_NM
            AND PKC.KEY_NM  = PK.KEY_NM
            AND PKC.USER_NM = PCOL.USER_NM
            AND PKC.RT_NM   = PCOL.RT_NM
            AND PKC.COL_NM  = PCOL.COL_NM
            AND NOT EXISTS
              (SELECT *
               FROM KEY_COLUMN FKC,
                    COLUMN     FCOL
               WHERE FKC.USER_NM       = FK.USER_NM
                 AND FKC.RT_NM         = FK.RT_NM
                 AND FKC.KEY_NM        = FK.KEY_NM
                 AND FKC.USER_NM       = FCOL.USER_NM
                 AND FKC.RT_NM         = FCOL.RT_NM
                 AND FKC.COL_NM        = FCOL.COL_NM
                 AND PKC.KEY_COL_SEQ#  = FKC.KEY_COL_SEQ#
                 AND FCOL.DOM_NM       = PCOL.DOM_NM))
      )
```

BM09- No two KEY-tuples, for which KEY_TYPE = 'PRIMARY', 'FOREIGN' or 'ALTERNATE', can be referenced by identical sets of KEY_COLUMN-tuples, except for key combinations in which the values of KEY_TYPE are either 'PRIMARY' and 'FOREIGN' or 'ALTERNATE' and 'FOREIGN'.

```
SELECT DISTINCT 'BM09: R-TABLE HAS TWO IDENTICAL KEYS (FK-FK, AK-AK, PK-AK)'
FROM KEY K1, KEY K2
WHERE K1.USER_NM = K2.USER_NM
  AND K1.RT_NM   = K2.RT_NM
  AND K1.KEY_NM <> K2.KEY_NM
  AND (
         K1.KEY_TYPE = K2.KEY_TYPE
       OR (
            K1.KEY_TYPE = 'PRIMARY'
        AND K2.KEY_TYPE = 'ALTERNATE'
           )
      )
  AND NOT EXISTS
     (SELECT * FROM KEY_COLUMN KC1
      WHERE KC1.USER_NM = K1.USER_NM
        AND KC1.RT_NM   = K1.RT_NM
        AND KC1.KEY_NM  = K1.KEY_NM
        AND NOT EXISTS
           (SELECT * FROM KEY_COLUMN KC2
            WHERE KC2.USER_NM = K2.USER_NM
              AND KC2.RT_NM   = K2.RT_NM
              AND KC2.KEY_NM  = K2.KEY_NM
              AND KC2.COL_NM  = KC1.COL_NM
           )
     )
  AND NOT EXISTS
     (SELECT * FROM KEY_COLUMN KC2
      WHERE KC2.USER_NM = K2.USER_NM
        AND KC2.RT_NM   = K2.RT_NM
        AND KC2.KEY_NM  = K2.KEY_NM
        AND NOT EXISTS
           (SELECT * FROM KEY_COLUMN KC1
            WHERE KC1.USER_NM = K1.USER_NM
              AND KC1.RT_NM   = K1.RT_NM
              AND KC1.KEY_NM  = K1.KEY_NM
              AND KC1.COL_NM  = KC2.COL_NM
           )
     )
```

BM10- The column RL_EXPR in R_TABLE contains an I-mark if and only if RT_TYPE = 'BASE'.

```
SELECT DISTINCT 'BM10: RL-EXPRESSION IS MISSING OR NOT ALLOWED'
FROM R_TABLE
WHERE (
         RT_TYPE = 'BASE'
       AND RL_EXPR IS NOT NULL
      )
   OR
      (    RT_TYPE <> 'BASE'
       AND RL_EXPR IS NULL
      )
```

BM11- The values of COL_SEQ# in the set of COLUMN-tuples with the same value for USER_NM and RT_NM are numbered consecutively up from 1 to the number of tuples in the set.

```
SELECT DISTINCT 'BM11: COLUMN-TUPLES MUST BE NUMBERED CONSECUTIVELY'
FROM COLUMN COL1
WHERE COL1.COL_SEQ# > 1
  AND NOT EXISTS
     (SELECT * FROM COLUMN COL2
      WHERE COL1.USER_NM  = COL2.USER_NM
        AND COL1.RT_NM    = COL2.RT_NM
        AND COL1.COL_SEQ# = COL2.COL_SEQ# + 1
     )
```

BM12- The values of KEY_COL_SEQ# in the set of KEY_COLUMN-tuples with the same value for USER_NM, RT_NM and KEY_NM are numbered consecutively up from 1 to the number of tuples in the set.

```
SELECT DISTINCT 'BM12: KEY_COLUMN-TUPLES MUST BE NUMBERED CONSECUTIVELY'
FROM KEY_COLUMN KC1
WHERE KC1.KEY_COL_SEQ# > 1
  AND NOT EXISTS
     (SELECT * FROM KEY_COLUMN KC2
      WHERE KC1.USER_NM      = KC2.USER_NM
        AND KC1.RT_NM        = KC2.RT_NM
        AND KC1.KEY_NM       = KC2.KEY_NM
        AND KC1.KEY_COL_SEQ# = KC2.KEY_COL_SEQ# + 1
     )
```

BM13- **No tuple in COLUMN that is referenced by a tuple in KEY_COLUMN that references a tuple in KEY having KEY_TYPE = 'PRIMARY' has a 'YES' value for either A_MARK_ALL or I_MARK_ALL.**

```
SELECT DISTINCT 'BM13: MARKS NOT ALLOWED FOR PRIMARY KEY COLUMNS'
FROM COLUMN
WHERE (
         A_MARK_ALL='YES'
       OR I_MARK_ALL='YES'
      )
  AND (USER_NM, RT_NM, COL_NM) IN
       (SELECT USER_NM, RT_NM, COL_NM
        FROM KEY_COLUMN
        WHERE (USER_NM, RT_NM, KEY_NM) IN
                (SELECT USER_NM, RT_NM, KEY_NM
                 FROM KEY
                 WHERE KEY_TYPE = 'PRIMARY'
                )
       )
```

BM14- **No tuple in COLUMN that is referenced by a tuple in KEY_COLUMN that references a tuple in KEY having KEY_TYPE = 'FOREIGN' has a 'YES' value for I_MARK_ALL.**

```
SELECT DISTINCT 'BM14: I-MARKS NOT ALLOWED FOR FOREIGN KEY COLUMNS'
FROM COLUMN
WHERE I_MARK_ALL='YES'
  AND (USER_NM, RT_NM, COL_NM) IN
       (SELECT USER_NM, RT_NM, COL_NM
        FROM KEY_COLUMN
        WHERE (USER_NM, RT_NM, KEY_NM) IN
                (SELECT USER_NM, RT_NM, KEY_NM
                 FROM KEY
                 WHERE KEY_TYPE = 'FOREIGN'
                )
      )
```

BM15- **Every value of LOVAL is less than or equal to the value of HIVAL in the same DOM_RANGE-tuple.**

```
SELECT DISTINCT 'BM15: LOWER BOUND OF DOMAIN RANGE EXCEEDS HIGHER BOUND'
FROM DOM_RANGE
WHERE LOVAL > HIVAL
```

BM16- **The values of DR_SEQ# in the set of DOM_RANGE-tuples with the same value for DOM_NM are numbered consecutively up from 1 to the number of tuples in the set.**

```
SELECT DISTINCT 'BM16: DOMAIN RANGE SEQUENCE MUST BE NUMBERED CONSECUTIVELY'
FROM DOM_RANGE DR1
WHERE DR1.DR_SEQ# > 1
  AND NOT EXISTS
     (SELECT * FROM DOM_RANGE DR2
      WHERE DR1.DOM_NM = DR2.DOM_NM
        AND DR1.DR_SEQ# = DR2.DR_SEQ# + 1
     )
```

BM17- **Tuples in DOM_RANGE with the same value for DOM_NM do not have overlapping ranges.**

```
SELECT DISTINCT 'BM17: DOMAIN RANGES MAY NOT OVERLAP'
FROM DOM_RANGE DR1,  /* lower range  */
     DOM_RANGE DR2   /* higher range */
WHERE DR1.DOM_NM  = DR2.DOM_NM
  AND DR1.DR_SEQ# = DR2.DR_SEQ# - 1
  AND DR1.HIVAL   > DR2.LOVAL
```

BM18- **Every value of LOVAL or HIVAL belongs to the basic data type of the DOMAIN-tuple referenced by the DOM_RANGE-tuple.**
This constraint cannot be expressed in RL.

3. COMPOSITE COLUMNS AND COMPOSITE DOMAINS

CM01- **For ever value of USER_NM, the set of tuples in COL_NM_SUP, projected over USER_NM, RT_NM and COL_NM_SUP, is disjunct from the set of tuples projected over USER_NM, RT_NM and COL_NM_SUB.**

```
SELECT DISTINCT 'CM01: MULTI-LEVEL COLUMN HIERARCHY NOT ALLOWED'
FROM COLUMN_STRUCT
WHERE (USER_NM, RT_NM, COL_NM_SUP) IN
        (SELECT USER_NM, RT_NM, COL_NM_SUB
         FROM COLUMN_STRUCT
        )
```

CM02- **The set of tuples in DOM_NM_SUP, projected over DOM_NM_SUP, is disjunct from the set of tuples projected over DOM_NM_SUB.**

```
SELECT DISTINCT 'CM02: MULTI-LEVEL DOMAIN HIERARCHY NOT ALLOWED'
FROM DOMAIN_STRUCT
WHERE DOM_NM_SUP IN
        (SELECT DOM_NM_SUB
         FROM DOMAIN_STRUCT
        )
```

CM03- **The values of CS_SEQ# in the set of COLUMN_STRUCT-tuples with the same value for COL_NM_SUP are numbered consecutively up from 1 to the number of tuples in the set.**

```
SELECT DISTINCT 'CM03: COLUMN STRUCTURE SEQUENCE MUST BE NUMBERED CONSECUTIVELY'
FROM COLUMN_STRUCT CS1
WHERE CS1.CS_SEQ# > 1
  AND NOT EXISTS
     (SELECT * FROM COLUMN_STRUCT CS2
      WHERE CS1.USER_NM     = CS2.USER_NM
        AND CS1.RT_NM       = CS2.RT_NM
        AND CS1.COL_NM_SUP  = CS2.COL_NM_SUP
        AND CS1.CS_SEQ#     = CS2.CS_SEQ# + 1
     )
```

CM04- **The values of DS_SEQ# in the set of DOMAIN_STRUCT-tuples with the same value for DOM_NM_SUP are numbered consecutively up from 1 to the number of tuples in the set.**

```
SELECT DISTINCT 'CM04: DOMAIN STRUCTURE SEQUENCE MUST BE NUMBERED CONSECUTIVELY'
FROM DOMAIN_STRUCT DS1
WHERE DS1.DS_SEQ# > 1
  AND NOT EXISTS
     (SELECT * FROM DOMAIN_STRUCT DS2
      WHERE DS1.DOM_NM_SUP = DS2.DOM_NM_SUP
        AND DS1.DS_SEQ#    = DS2.DS_SEQ# + 1
     )
```

CM05- **For every composite COLUMN-tuple there exist corresponding pairs of COLUMN_STRUCT and DOMAIN_STRUCT-tuple for which:**

- **CS_SEQ# and DS_SEQ# are equal;**
- **DOM_NM_SUP in DOM_STRUCT equals DOM_NM in the COLUMN-tuple designated by COL_NM_SUP;**
- **DOM_NM_SUB in DOM_STRUCT equals DOM_NM in the COLUMN-tuple referenced by the COLUMN_STRUCT-tuple.**

```
SELECT DISTINCT 'CM05: COMPOSITE COLUMN HAS NO MATCHING COMPOSITE DOMAIN'
FROM COLUMN CSUP
WHERE EXISTS
     (SELECT *            /* column_struct without matching domain_struct */
      FROM COLUMN_STRUCT CS,
           COLUMN        CSUB
      WHERE CS.USER_NM    = CSUP.USER_NM
        AND CS.RT_NM      = CSUP.RT_NM
        AND CS.COL_NM_SUP = CSUP.COL_NM
        AND CS.USER_NM    = CSUB.USER_NM
        AND CS.RT_NM      = CSUB.RT_NM
        AND CS.COL_NM_SUB = CSUB.COL_NM
        AND NOT EXISTS
            (SELECT * FROM DOMAIN_STRUCT DS
             WHERE DS.DOM_NM_SUP = CSUP.DOM_NM
               AND DS.DOM_NM_SUB = CSUB.DOM_NM
               AND DS.DS_SEQ#    = CS.CS_SEQ#
            )
     )
   OR EXISTS
     (SELECT *            /* domain_struct without matching column_struct */
      FROM DOMAIN_STRUCT DS
      WHERE CSUP.DOM_NM = DS.DOM_NM_SUP
        AND NOT EXISTS
           (SELECT *
            FROM COLUMN        CSUB,
                 COLUMN_STRUCT CS
            WHERE CS.USER_NM    = CSUB.USER_NM
              AND CS.RT_NM      = CSUB.RT_NM
              AND CS.COL_NM_SUB = CSUB.COL_NM
              AND CS.USER_NM    = CSUP.USER_NM
              AND CS.RT_NM      = CSUP.RT_NM
              AND CS.COL_NM_SUP = CSUP.COL_NM
              AND CSUB.DOM_NM   = DS.DOM_NM_SUB
              AND CS.CS_SEQ#    = DS.DS_SEQ#
           )
     )
```

CM06- **If columns constituting a composite column have different values for A_MARK_ALL or I_MARK_ALL, these columns are assigned I_MARKs for the composite column tuple, while the values are inherited by the composite column tuple if the elementary columns have similar values for A_MARK_ALL or I_MARK_ALL. Elementary columns cannot accept marks for A_MARK_ALL and I_MARK_ALL.**

```
SELECT DISTINCT 'CM06: INCONSISTENCY BETWEEN MARKS FOR ELEMENTARY AND COMPOSITE COLUMNS'
FROM COLUMN CSUP
WHERE (
          A_MARK_ALL IS NULL   /* i-mark */
      AND 2 >
          (SELECT COUNT(DISTINCT CSUB.A_MARK_ALL)
           FROM COLUMN         CSUB,
                COLUMN_STRUCT CS
           WHERE CS.USER_NM     = CSUB.USER_NM
             AND CS.RT_NM       = CSUB.RT_NM
             AND CS.COL_NM_SUB = CSUB.COL_NM
             AND CS.USER_NM     = CSUP.USER_NM
             AND CS.RT_NM       = CSUP.RT_NM
             AND CS.COL_NM_SUP = CSUP.COL_NM
          )
   )
OR
   (
          I_MARK_ALL IS NULL   /* i-mark */
      AND 2 >
          (SELECT COUNT(DISTINCT CSUB.I_MARK_ALL)
           FROM COLUMN         CSUB,
                COLUMN_STRUCT CS
           WHERE CS.USER_NM     = CSUB.USER_NM
             AND CS.RT_NM       = CSUB.RT_NM
             AND CS.COL_NM_SUB = CSUB.COL_NM
             AND CS.USER_NM     = CSUP.USER_NM
             AND CS.RT_NM       = CSUP.RT_NM
             AND CS.COL_NM_SUP = CSUP.COL_NM
          )
   )
OR
   (
          A_MARK_ALL IS NOT NULL  /* not i-marked */
      AND EXISTS
          (SELECT *
           FROM COLUMN         CSUB,
                COLUMN_STRUCT CS
           WHERE CS.USER_NM     = CSUB.USER_NM
             AND CS.RT_NM       = CSUB.RT_NM
             AND CS.COL_NM_SUB = CSUB.COL_NM
             AND CS.USER_NM     = CSUP.USER_NM
             AND CS.RT_NM       = CSUP.RT_NM
             AND CS.COL_NM_SUP = CSUP.COL_NM
             AND CSUB.A_MARK_ALL <> CSUP.A_MARK_ALL
          )
   )
OR
   (
          I_MARK_ALL IS NOT NULL  /* not i-marked */
      AND EXISTS
          (SELECT *
           FROM COLUMN         CSUB,
                COLUMN_STRUCT CS
           WHERE CS.USER_NM     = CSUB.USER_NM
             AND CS.RT_NM       = CSUB.RT_NM
             AND CS.COL_NM_SUB = CSUB.COL_NM
             AND CS.USER_NM     = CSUP.USER_NM
             AND CS.RT_NM       = CSUP.RT_NM
             AND CS.COL_NM_SUP = CSUP.COL_NM
             AND CSUB.I_MARK_ALL <> CSUP.I_MARK_ALL
          )
   )
```

CM07- If domains constituting a composite domain have different values for COMP_ALL, this column is assigned the value 'NO' for the composite domain tuple, while the value is inherited by the composite domain if the elementary domains do not have different values for COMP_ALL.

```
SELECT DISTINCT 'CM07: INCONSISTENCY BETWEEN COMP_ALL FOR ELEMENTARY AND COMPOSITE DOMAINS'
FROM DOMAIN DSUP
WHERE (
          COMP_ALL = 'YES'
      AND EXISTS
          (SELECT *
           FROM DOMAIN        DSUB,
                DOMAIN_STRUCT DS
           WHERE DS.DOM_NM_SUB = DSUB.DOM_NM
             AND DS.DOM_NM_SUP = DSUP.DOM_NM
             AND DSUB.COMP_ALL = 'NO'
          )
    )
  OR
    (
          COMP_ALL = 'NO'
      AND EXISTS
          (SELECT * FROM DOMAIN_STRUCT DS
           WHERE DS.DOM_NM_SUP = DSUP.DOM_NM
          )
      AND 1 =
          (SELECT COUNT(DISTINCT DSUB.COMP_ALL)
           FROM DOMAIN        DSUB,
                DOMAIN_STRUCT DS
           WHERE DS.DOM_NM_SUB = DSUB.DOM_NM
             AND DS.DOM_NM_SUP = DSUP.DOM_NM
             AND DSUB.COMP_ALL = 'YES'
          )
    )
```

CM08- DOMAIN-tuples must contain an A-mark for BD_TYPE_NM and I-marks for LENGTH and SCALE if and only if they are referenced by DOMAIN_STRUCT-tuples by means of the column DOM_NM_SUP.

```
SELECT DISTINCT 'CM08: ELEMENTARY DOMAIN HAS NO DATATYPE DATA OR COMPOSITE DOMAIN HAS'
FROM DOMAIN
WHERE
     (    BD_TYPE_NM IS NULL    /* a-marked */
      AND DOM_NM NOT IN
            (SELECT DOM_NM_SUP
             FROM DOMAIN_STRUCT
            )
    )
  OR
    (    BD_TYPE_NM IS NOT NULL    /* not a-marked */
      AND DOM_NM IN
            (SELECT DOM_NM_SUP
             FROM DOMAIN_STRUCT
            )
    )
  OR
    (    BD_TYPE_NM IS NULL    /* a-marked */
      AND (LENGTH IS NOT NULL OR SCALE IS NOT NULL)  /* not i-marked */
    )
```

CM09- A KEY_COLUMN-tuple must not reference a composite COLUMN-tuple.

```
SELECT DISTINCT 'CM09: COMPOSITE COLUMN MUST NOT BE A COLUMN CONSTITUTING A KEY'
FROM COLUMN
WHERE (USER_NM, RT_NM, COL_NM) IN
       (SELECT USER_NM, RT_NM, COL_NM_SUP
        FROM COLUMN_STRUCT
       )
  AND (USER_NM, RT_NM, COL_NM) IN
       (SELECT USER_NM, RT_NM, COL_NM
        FROM KEY_COLUMN
       )
```

4. ARCHIVES, SNAPSHOTS AND RELATIONAL ASSIGNMENTS

AR01- The combination of CR_DATE and CR_TIME in ARCHIVE does not exceed the current system date and time.

```
SELECT DISTINCT 'AR01: ARCHIVE CREATED AFTER SYSTEM DATE'  /* oracle-specific statement */
FROM ARCHIVE
WHERE TO_CHAR(SYSDATE, 'YYYYMMDDHH24MMSS') < TO_CHAR(CR_DATE, 'YYYYMMDD') || CR_TIME
```

AR02- The combination of CR_DATE and CR_TIME in RT_SITE does not exceed the current system date and time.

```
SELECT DISTINCT 'AR02: R-TABLE CREATED AFTER SYSTEM DATE' /* oracle-specific statement */
FROM RT_SITE
WHERE TO_CHAR(SYSDATE, 'YYYYMMDDHH24MMSS') < TO_CHAR(CR_DATE, 'YYYYMMDD') || CR_TIME
```

AR03- Tuples in R_TABLE for which RT_TYPE <> 'BASE', 'VIEW' or 'CHECKOUT' are not referenced by KEY-tuples for which KEY_TYPE = 'FOREIGN'.

```
SELECT DISTINCT 'AR03: NON BASE R-TABLE, NON-VIEW, NON-CHECKOUT HAS FOREIGN KEY'
FROM R_TABLE
WHERE RT_TYPE NOT IN ('BASE', 'VIEW', 'CHECKOUT')
  AND (USER_NM, RT_NM) IN
        (SELECT USER_NM, RT_NM
         FROM KEY
         WHERE KEY_TYPE = 'FOREIGN'
        )
```

5. VIEWS

VW01- An R_TABLE-tuple contains I-marked values for INS_ALL and DEL_ALL and a COLUMN-tuple contains I-marked values for UPD_ALL if and only if RT_TYPE <> 'VIEW'.

```
SELECT DISTINCT 'VW01: VIEW HAS MARKS FOR INS_ALL, UPD_ALL OR DEL_ALL'
FROM R_TABLE
WHERE (    RT_TYPE = 'VIEW'
       AND (INS_ALL IS NULL OR DEL_ALL IS NULL)  /* i-marked */
      )
   OR
      (    RT_TYPE = 'VIEW'
       AND (USER_NM, RT_NM) IN
            (SELECT USER_NM, RT_NM
             FROM COLUMN
             WHERE UPD_ALL IS NULL   /* i-marked */
            )
      )
   OR
      (    RT_TYPE <> 'VIEW'
       AND (INS_ALL IS NOT NULL OR DEL_ALL IS NOT NULL)  /* not i-marked */
      )
   OR
      (    RT_TYPE <> 'VIEW'
       AND (USER_NM, RT_NM) IN
            (SELECT USER_NM, RT_NM
             FROM COLUMN
             WHERE UPD_ALL IS NOT NULL   /* not i-marked */
            )
      )
```

VW02- For every COLUMN-tuple, the columns USER_NM, RT_NM_BASE and COL_NM_BASE are both I-marked or not marked.

```
SELECT DISTINCT 'VW02: USER_NM_BASE, RT_NM_BASE AND COL_NM_BASE ARE PARTLY A-MARKED'
FROM COLUMN
WHERE (    USER_NM_BASE IS NULL  /* a-marked */
       AND (RT_NM_BASE IS NOT NULL OR COL_NM_BASE IS NOT NULL)
      )
   OR (    RT_NM_BASE IS NULL    /* a-marked */
       AND (USER_NM_BASE IS NOT NULL OR COL_NM_BASE IS NOT NULL)
      )
   OR (    COL_NM_BASE IS NULL   /* a-marked */
       AND (USER_NM_BASE IS NOT NULL OR RT_NM_BASE IS NOT NULL)
      )
```

VW03- Every COLUMN-tuple for which the column RT_NM_BASE is not I-marked, references an R_TABLE-tuple for which RT_TYPE contains the value 'VIEW'.

```
SELECT DISTINCT 'VW03: NON-VIEW COLUMN REFERENCES ANOTHER COLUMN'
FROM COLUMN
WHERE RT_NM_BASE IS NOT NULL  /* not a-marked */
  AND (USER_NM, RT_NM) NOT IN
        (SELECT USER_NM, RT_NM
         FROM R_TABLE
         WHERE RT_TYPE = 'VIEW'
        )
```

VW04- COLUMN-tuples for which the column DOM_NM is A-marked reference R_TABLE-tuples for which RT_TYPE = 'VIEW', 'SNAPSHOT' or 'REL_ASGN'.

```
SELECT DISTINCT 'VW04: DOM_NM CAN ONLY BE MARKED FOR VIEWS, SNAPSHOTS AND REL.ASSIGNMENTS'
FROM COLUMN
WHERE DOM_NM IS NULL  /* a-marked */
  AND (USER_NM, RT_NM) NOT IN
        (SELECT USER_NM, RT_NM
         FROM R_TABLE
         WHERE RT_TYPE IN ('VIEW', 'SNAPSHOT', 'REL_ASGN')
        )
```

VW05- COLUMN-tuples, for which the column DOM_NM is not marked, have values for BD_TYPE_NM, LENGTH, and SCALE that are equal to the values of similarly named columns in the referenced DOMAIN-tuple.

```
SELECT DISTINCT 'VW05: DATATYPE DATA OF COLUMN AND DOMAIN DO NOT MATCH'
FROM COLUMN  COL,
     DOMAIN  DOM
WHERE COL.DOM_NM = DOM.DOM_NM
  AND (   COL.BD_TYPE_NM <> DOM.BD_TYPE_NM
       OR COL.LENGTH     <> DOM.LENGTH
       OR COL.SCALE      <> DOM.SCALE
      )
```

6. INDEXES

IX01- The projections of DB_INDEX and KEY on their columns USER_NM and INDEX_NM are disjunct.

```
SELECT DISTINCT 'IX01: CONVENTIONAL INDEX NAME COINCIDES WITH DOMAIN-BASED INDEX NAME'
FROM KEY
WHERE (USER_NM, INDEX_NM) IN
        (SELECT USER_NM, INDEX_NM
         FROM DB_INDEX
        )
```

IX02- Every DB_INDEX-tuple is referenced by at least two DB_INDEX_COL-tuples.

```
SELECT DISTINCT 'IX02: DOMAIN-BASED INDEX MUST CONSIST OF AT LEAST 2 COLUMNS'
FROM DB_INDEX DBI
WHERE 2 >
        (SELECT COUNT(*)
         FROM DB_INDEX_COL DBICOL
         WHERE DBICOL.USER_NM_DBI = DBI.USER_NM
           AND DBICOL.INDEX_NM    = DBI.INDEX_NM
        )
```

IX03- Every tuple in DB_INDEX_COL references the same DOMAIN-tuple via COLUMN and DB_INDEX.

```
SELECT DISTINCT 'IX03: COLUMNS IN DOMAIN-BASED INDEX DO NOT BELONG TO THE DOMAIN OF THE INDEX'
FROM DB_INDEX DBI
WHERE (USER_NM, INDEX_NM) IN
        (SELECT USER_NM_DBI, INDEX_NM
         FROM DB_INDEX_COL DBICOL,
              COLUMN       COL
         WHERE DBICOL.USER_NM_RT  = COL.USER_NM
           AND DBICOL.RT_NM       = COL.RT_NM
           AND DBICOL.COL_NM      = COL.COL_NM
           AND DBICOL.USER_NM_DBI = DBI.USER_NM
           AND DBICOL.INDEX_NM    = DBI.INDEX_NM
           AND DBI.DOM_NM        <> COL.DOM_NM
        )
```

IX04- The projection of the columns USER_NM_RT, RT_NM and COL_NM in DB_INDEX_COL for one given value of INDEX_NM is not a subset of another such projection for another value of INDEX_NM.

```
SELECT DISTINCT 'IX04: DOMAIN-BASED INDEX IS REDUNDANT'
FROM DB_INDEX DBI_1,    /* non-redundant index */
     DB_INDEX DBI_2     /* redundant index     */
WHERE DBI_1.DOM_NM = DBI_2.DOM_NM
  AND (   DBI_1.USER_NM  <> DBI_2.USER_NM
       OR DBI_1.INDEX_NM <> DBI_2.INDEX_NM
      )
  AND NOT EXISTS
      (SELECT * FROM DB_INDEX_COL DBICOL_1  /* column of redundant index */
       WHERE DBICOL_1.USER_NM_DBI = DBI_1.USER_NM
         AND DBICOL_1.INDEX_NM    = DBI_1.INDEX_NM
         AND NOT EXISTS
             (SELECT * FROM DB_INDEX_COL DBICOL_2  /* column of non-redundant index */
              WHERE DBICOL_2.USER_NM_DBI = DBI_2.USER_NM
                AND DBICOL_2.INDEX_NM    = DBI_2.INDEX_NM
                AND DBICOL_2.USER_NM_RT  = DBICOL_1.USER_NM_RT
                AND DBICOL_2.RT_NM       = DBICOL_1.RT_NM
                AND DBICOL_2.COL_NM      = DBICOL_1.COL_NM
             )
      )
```

IX05- DB_INDEX_COL-tuples do not reference R_TABLE-tuples (via COLUMN) that have the value 'VIEW' for their column RT_TYPE.

```
SELECT DISTINCT 'IX05: COLUMN IN DOMAIN-BASED INDEX MUST NOT BE A VIEW-COLUMN'
FROM DB_INDEX_COL
WHERE (USER_NM_RT, RT_NM) IN
        (SELECT USER_NM, RT_NM
         FROM R_TABLE
         WHERE RT_TYPE = 'VIEW'
        )
```

IX06- KEY-tuples for which the column INDEX_NM is not I-marked reference R_TABLE-tuples for which RT_TYPE <> 'VIEW'.

```
SELECT DISTINCT 'IX06: CONVENTIONAL INDEX CANNOT NOT BE DEFINED ON A VIEW'
FROM KEY
WHERE INDEX_NM IS NOT NULL    /* a-marked */
  AND (USER_NM, RT_NM) IN
       (SELECT USER_NM, RT_NM
        FROM R_TABLE
        WHERE RT_TYPE = 'VIEW'
       )
```

IX07- KEY-tuples for which KEY_TYPE = 'INDEX' do not have a I-mark for their column INDEX_NM.

```
SELECT DISTINCT 'IX07: INDEX KEYS MUST HAVE A NAME'
FROM KEY
WHERE KEY_TYPE = 'INDEX'
  AND INDEX_NM IS NULL   /* a-marked */
```

IX08- For the subset of KEY-tuples for which INDEX_NM is not I-marked, the projection of the columns USER_NM, RT_NM, COL_NM and KEY_COL_SEQ# in KEY_COLUMN for one given value of KEY_NM is not a subset of another such projection for another KEY-tuple.

```
SELECT DISTINCT 'IX08: CONVENTIONAL INDEX IS REDUNDANT'
FROM KEY KEY_1,   /* non-redundant index */
     KEY KEY_2    /* redundant index     */
WHERE KEY_1.USER_NM   = KEY_2.USER_NM
  AND KEY_1.RT_NM     = KEY_2.RT_NM
  AND KEY_1.INDEX_NM <> KEY_2.INDEX_NM
  AND NOT EXISTS
      (SELECT * FROM KEY_COLUMN KC_1         /* column of redundant index */
       WHERE KC_1.USER_NM = KEY_1.USER_NM
         AND KC_1.RT_NM   = KEY_1.RT_NM
         AND KC_1.KEY_NM  = KEY_1.KEY_NM
         AND NOT EXISTS
             (SELECT * FROM KEY_COLUMN KC_2  /* column of non-redundant index */
              WHERE KC_2.USER_NM      = KEY_2.USER_NM
                AND KC_2.RT_NM        = KEY_2.RT_NM
                AND KC_2.KEY_NM       = KEY_2.KEY_NM
                AND KC_2.COL_NM       = KC_1.COL_NM
                AND KC_2.KEY_COL_SEQ# = KC_1.KEY_COL_SEQ#
             )
      )
```

7. CONSTRAINTS AND TRIGGERED ACTIONS

CA01- A COND_ACT-tuple can be referenced by a tuple in KEY, DOMAIN, COLUMN, RT_CA or COL_CA if and only if the column CONDITION is not I-marked.

```
SELECT DISTINCT 'CA01: CONDITION IS WRONGLY I-MARKED OR NOT I-MARKED'
FROM COND_ACT
WHERE (    CONDITION IS NULL   /* i-marked */
       AND (USER_NM, CA_NM) IN
             (SELECT USER_NM,    CA_NM   FROM KEY
              UNION
              SELECT USER_NM,    CA_NM   FROM DOMAIN
              UNION
              SELECT USER_NM,    CA_NM_C FROM COLUMN
              UNION
              SELECT USER_NM,    CA_NM_M FROM COLUMN
              UNION
              SELECT USER_NM_CA, CA_NM   FROM RT_CA
              UNION
              SELECT USER_NM_CA, CA_NM   FROM COL_CA
             )
      )
  OR
      (    CONDITION IS NOT NULL   /* not i-marked */
       AND (USER_NM, CA_NM) NOT IN
             (SELECT USER_NM,    CA_NM   FROM KEY
              WHERE CA_NM IS NOT NULL  /* a-marked */
              UNION
              SELECT USER_NM,    CA_NM   FROM DOMAIN
              WHERE CA_NM IS NOT NULL  /* a-marked */
              UNION
              SELECT USER_NM,    CA_NM_C FROM COLUMN
              WHERE CA_NM_C IS NOT NULL  /* a-marked */
              UNION
              SELECT USER_NM,    CA_NM_M FROM COLUMN
              WHERE CA_NM_M IS NOT NULL  /* a-marked */
              UNION
              SELECT USER_NM_CA, CA_NM   FROM RT_CA
              UNION
              SELECT USER_NM_CA, CA_NM   FROM COL_CA
             )
      )
```

CA02- If a COND_ACT-tuple is referenced by a KEY-tuple, then it must not be referenced by a tuple in DOMAIN, COLUMN, RT_CA or COL_CA.

```
SELECT DISTINCT 'CA02: CONDITIONAL ACTION IS IMPROPERLY CLASSIFIED'
FROM COND_ACT
WHERE (USER_NM, CA_NM) IN
        (SELECT USER_NM, CA_NM
         FROM KEY
        )
  AND (USER_NM, CA_NM) IN
        (SELECT USER_NM,    CA_NM   FROM DOMAIN
         UNION
         SELECT USER_NM,    CA_NM_C FROM COLUMN
         UNION
         SELECT USER_NM,    CA_NM_M FROM COLUMN
         UNION
         SELECT USER_NM_CA, CA_NM   FROM RT_CA
         UNION
         SELECT USER_NM_CA, CA_NM   FROM COL_CA
        )
```

CA03- If a COND_ACT-tuple is referenced by a DOMAIN-tuple, then it must not be referenced by a tuple in KEY, COLUMN, RT_CA or COL_CA.

```
SELECT DISTINCT 'CA03: CONDITIONAL ACTION IS IMPROPERLY CLASSIFIED'
FROM COND_ACT
WHERE (USER_NM, CA_NM) IN
        (SELECT USER_NM, CA_NM
         FROM DOMAIN
        )
  AND (USER_NM, CA_NM) IN
        (SELECT USER_NM,    CA_NM   FROM KEY
         UNION
         SELECT USER_NM,    CA_NM_C FROM COLUMN
         UNION
         SELECT USER_NM,    CA_NM_M FROM COLUMN
         UNION
         SELECT USER_NM_CA, CA_NM   FROM RT_CA
         UNION
         SELECT USER_NM_CA, CA_NM   FROM COL_CA
        )
```

CA04- If a COND_ACT-tuple is referenced by a COLUMN-tuple by means of its column CA_NM_C, then it must not be referenced by a tuple in KEY, DOMAIN, COLUMN, RT_CA or COL_CA.

```
SELECT DISTINCT 'CA04: CONDITIONAL ACTION IS IMPROPERLY CLASSIFIED'
FROM COND_ACT
WHERE (USER_NM, CA_NM) IN
        (SELECT USER_NM, CA_NM_C
         FROM COLUMN
        )
  AND (USER_NM, CA_NM) IN
        (SELECT USER_NM,    CA_NM   FROM KEY
         UNION
         SELECT USER_NM,    CA_NM   FROM DOMAIN
         UNION
         SELECT USER_NM,    CA_NM_M FROM COLUMN
         UNION
         SELECT USER_NM_CA, CA_NM   FROM RT_CA
         UNION
         SELECT USER_NM_CA, CA_NM   FROM COL_CA
        )
```

CA05- If a COND_ACT-tuple is referenced by a COLUMN-tuple by means of its column CA_NM_M, then it must not be referenced by a tuple in KEY, DOMAIN, COLUMN, RT_CA or COL_CA.

```
SELECT DISTINCT 'CA05: CONDITIONAL ACTION IS IMPROPERLY CLASSIFIED'
FROM COND_ACT
WHERE (USER_NM, CA_NM) IN
        (SELECT USER_NM, CA_NM_M
         FROM COLUMN
        )
  AND (USER_NM, CA_NM) IN
        (SELECT USER_NM,    CA_NM   FROM KEY
         UNION
         SELECT USER_NM,    CA_NM   FROM DOMAIN
         UNION
         SELECT USER_NM,    CA_NM_C FROM COLUMN
         UNION
         SELECT USER_NM_CA, CA_NM   FROM RT_CA
         UNION
         SELECT USER_NM_CA, CA_NM   FROM COL_CA
        )
```

CA06- **If a COND_ACT-tuple is referenced by one or more tuples in RT_CA or COL_CA, then it must not be referenced by a tuple in KEY, DOMAIN or COLUMN.**

```
SELECT DISTINCT 'CA06: CONDITIONAL ACTION IS IMPROPERLY CLASSIFIED'
FROM COND_ACT
WHERE (USER_NM, CA_NM) IN
        (SELECT USER_NM_CA, CA_NM
         FROM RT_CA
         UNION
         SELECT USER_NM_CA, CA_NM
         FROM COL_CA
        )
  AND (USER_NM, CA_NM) IN
        (SELECT USER_NM, CA_NM   FROM KEY
         UNION
         SELECT USER_NM, CA_NM   FROM DOMAIN
         UNION
         SELECT USER_NM, CA_NM_C FROM COLUMN
         UNION
         SELECT USER_NM, CA_NM_M FROM COLUMN
        )
```

CA07- **The column CA_NM_M in COLUMN must be A-marked if and only if both A_MARK_ALL and I_MARK_ALL contain the value 'YES'.**

```
SELECT DISTINCT 'CA07: MISSING OR SUPERFLUOUS MARK-CONSTRAINT'
FROM COLUMN
WHERE (    CA_NM_M IS NULL   /* a-marked */
       AND (   A_MARK_ALL = 'NO'
            OR I_MARK_ALL = 'NO'
           )
      )
   OR (    CA_NM_M IS NOT NULL   /* not a-marked */
       AND A_MARK_ALL = 'YES'
       AND I_MARK_ALL = 'YES'
      )
```

CA08- **Preferably, no COND_ACT-tuple that is referenced by either a DOMAIN-tuple (D-constraint) or a COLUMN-tuple (C-constraint) has a value 'TT' for TTIME.**

```
SELECT DISTINCT 'CA08: (WARNING) CONSTRAINT OF TYPE C OR D SHOULD NORMALLY BE TC-TIMED'
FROM COND_ACT
WHERE TTIME = 'TT'
  AND (USER_NM, CA_NM) IN
        (SELECT USER_NM, CA_NM_C
         FROM COLUMN
         UNION
         SELECT USER_NM, CA_NM
         FROM DOMAIN
        )
```

CA09- **Preferably, no tuple in COND_ACT, for which CONDITION is not I-marked, references tuples in R_TABLE having RT_TYPE <> 'BASE' via RT_CA, COL_CA, COLUMN, or KEY having KEY_TYPE not equal to 'PRIMARY'.**

```
SELECT DISTINCT 'CA09: (WARNING) NON-E CONSTRAINTS ARE NORMALLY APPLIED TO BASE R-TABLES ONLY'
FROM COND_ACT CA
WHERE CONDITION IS NOT NULL  /* not i-marked */
  AND EXISTS
        (SELECT * FROM R_TABLE RT
         WHERE RT_TYPE <> 'BASE'
           AND (USER_NM, RT_NM) IN
                 (SELECT USER_NM, RT_NM
                  FROM KEY
                  WHERE KEY_TYPE <> 'PRIMARY'
                    AND CA_NM    =  CA.CA_NM
                  UNION
                  SELECT USER_NM, RT_NM
                  FROM DOMAIN
                  WHERE CA_NM = CA.CA_NM
                  UNION
                  SELECT USER_NM, RT_NM
                  FROM COLUMN
                  WHERE CA_NM_C = CA.CA_NM
                     OR CA_NM_C = CA.CA_NM
                  UNION
                  SELECT USER_NM_RT, RT_NM
                  FROM RT_CA
                  WHERE USER_NM_CA = CA.USER_NM
                    AND CA_NM      = CA.CA_NM
                  UNION
                  SELECT USER_NM_RT, RT_NM
                  FROM COL_CA
                  WHERE USER_NM_CA = CA.USER_NM
                    AND CA_NM      = CA.CA_NM
                 )
        )
```

CA10- If INTERVAL in COND_ACT does not contain an I-mark, CA_DATE and CA_TIME may not contain such a mark either.

```
SELECT DISTINCT 'CA10: AN INTERVAL HAS BEEN SPECIFIED WITHOUT A STARTING DATE AND TIME'
FROM COND_ACT CA
WHERE INTERVAL IS NOT NULL  /* not i-marked */
  AND (   CA_DATE IS NULL   /* i-marked     */
       OR CA_TIME IS NULL   /* i-marked     */
      )
```

CA11- If a COND_ACT-tuple designates a dynamic constraint (IND_STATIC = 'NO'), then the column CONDITION must not be I-marked, and the columns CA_DATE and CONSTR_TYPE must be I-marked,

```
SELECT DISTINCT 'CA11: INCORRECT SPECIFICATION OF DYNAMIC CONSTRAINT'
FROM COND_ACT
WHERE     IND_STATIC = 'NO'
      AND (   CONDITION   IS NULL       /*     i-marked */
           OR CA_DATE     IS NOT NULL   /* not i-marked */
           OR CONSTR_TYPE IS NOT NULL   /* not i-marked */
          )
```

CA12- The columns CA_DATE and CA_TIME in COND_ACT are either both I-marked or not marked.

```
SELECT DISTINCT 'CA12: COLUMNS CA_DATE AND CA_TIME MUST BOTH BE MARKED OR NOT MARKED'
FROM COND_ACT CA
WHERE (    CA_DATE IS NULL      /*     i-marked */
       AND CA_TIME IS NOT NULL  /* not i-marked */
      )
   OR (    CA_DATE IS NOT NULL  /* not i-marked */
       AND CA_TIME IS NULL      /*     i-marked */
      )
```

CA13- The columns CONDITION and TTIME in COND_ACT are either both I-marked or not marked.

```
SELECT DISTINCT 'CA13: COLUMNS CONDITION AND TTIME MUST BOTH BE MARKED OR NOT MARKED'
FROM COND_ACT CA
WHERE (    CONDITION IS NULL      /*     i-marked */
       AND TTIME     IS NOT NULL  /* not i-marked */
      )
   OR (    CONDITION IS NOT NULL  /* not i-marked */
       AND TTIME     IS NULL      /*     i-marked */
      )
```

CA14- The column CA_NM in KEY must be A-marked if and only if KEY_TYPE = 'INDEX'.

```
SELECT DISTINCT 'CA14: CONDITIONAL ACTION CANNOT BE APPLIED TO A CONVENTIONAL INDEX'
FROM KEY
WHERE KEY_TYPE = 'INDEX'
  AND CA_NM IS NOT NULL
```

CA15- Every COLUMN-tuple that belongs to a primary key has an I-mark for CA_NM_M.

```
SELECT DISTINCT 'CA15: MARK-CONSTRAINT FOR PRIMARY KEY COLUMN IS SUPERFLUOUS'
FROM COLUMN
WHERE CA_NM_M IS NOT NULL   /* not a-marked */
  AND (USER_NM, RT_NM, COL_NM) IN
       (SELECT USER_NM, RT_NM, COL_NM
        FROM KEY_COLUMN
        WHERE (USER_NM, RT_NM, KEY_NM) IN
                (SELECT USER_NM, RT_NM, KEY_NM
                 FROM KEY
                 WHERE KEY_TYPE = 'PRIMARY'
                )
       )
```

CA16- The columns UPD_ACT and DEL_ACT in KEY must be I-marked if and only if KEY_TYPE <> 'FOREIGN'.

```
SELECT DISTINCT 'CA16: ACTION ON FOREIGN KEY UPDATE/DELETE IS ONLY RELEVANT FOR FOREIGN KEYS'
FROM KEY
WHERE (    KEY_TYPE = 'FOREIGN'
       AND (   UPD_ACT IS NULL    /* i-marked */
            OR DEL_ACT IS NULL    /* i-marked */
           )
      )
   OR (    KEY_TYPE <> 'FOREIGN'
       AND (   UPD_ACT IS NOT NULL    /* not i-marked */
            OR DEL_ACT IS NOT NULL    /* not i-marked */
           )
      )
```

CA17- Tuples in KEY for which KEY_TYPE = 'PRIMARY' reference tuples in COND_ACT for which TTIME = 'TC'.

```
SELECT DISTINCT 'CA17: E-TYPE CONSTRAINT MUST BE TC-TIMED'
FROM KEY
WHERE KEY_TYPE = 'PRIMARY'
  AND (USER_NM, CA_NM) NOT IN
       (SELECT USER_NM, CA_NM
        FROM COND_ACT
        WHERE TTIME = 'TC'
       )
```

CA18- **Tuples in COND_ACT for which CONSTR_TYPE is not I-marked cannot accept I-marks for CONDITION.**

```
SELECT DISTINCT 'CA18: TYPED CONSTRAINT CANNOT HAVE AN I-MARK FOR CONDITION COLUMN'
FROM COND_ACT
WHERE CONSTR_TYPE IS NOT NULL  /* not i-marked */
  AND CONDITION IS NULL        /*     i-marked */
```

CA19- **The column GROUP_ID in COL_CA must be I-marked if and only if the column CONSTR_TYPE in COND_ACT is also I-marked.**

```
SELECT DISTINCT 'CA19: COLUMN GROUP_ID MUST BE I-MARKED OR UNMARKED'
FROM COND_ACT
WHERE (    CONSTR_TYPE IS NOT NULL  /* not i-marked */
       AND (USER_NM, CA_NM) IN
             (SELECT USER_NM, CA_NM
              FROM COL_CA
              WHERE GROUP_ID IS NULL  /* i-marked */
             )
      )
   OR (    CONSTR_TYPE IS NULL  /* i-marked */
       AND (USER_NM, CA_NM) IN
             (SELECT USER_NM, CA_NM
              FROM COL_CA
              WHERE GROUP_ID IS NOT NULL  /* not i-marked */
             )
      )
```

CA20- **Tuples in COND_ACT for which CONSTR_TYPE = 'FD' are referenced by at least one COL_CA-tuple for which GROUP_ID = 'DETERMINING', by at least one COL_CA-tuple with GROUP_ID = 'DEPENDENT', by no other tuples in COL_CA, by one tuple in RT_CA, and by no tuple in any other R-table. The columns refer to the same R-table and the RT_CA-tuple has the value 'YES' values for the column INS 'No' values for the columns 'DEL' and 'RET'.**

```
SELECT DISTINCT 'CA20: INCORRECT SPECIFICATION OF FUNCTIONAL DEPENDENCY CONSTRAINT'
FROM COND_ACT CA
WHERE CONSTR_TYPE = 'FD'
  AND (
        (USER_NM, CA_NM) NOT IN
          (SELECT USER_NM, CA_NM
           FROM COL_CA
           WHERE GROUP_ID = 'DETERMINING'
          )
       OR (USER_NM, CA_NM) NOT IN
          (SELECT USER_NM, CA_NM
           FROM COL_CA
           WHERE GROUP_ID = 'DEPENDENT'
          )
       OR (USER_NM, CA_NM) IN
          (SELECT USER_NM, CA_NM
           FROM COL_CA
           WHERE GROUP_ID NOT IN ('DETERMINING','DEPENDENT')
          )
       OR 1 <>
          (SELECT COUNT(*)
           FROM RT_CA
           WHERE USER_NM_CA = CA.USER_NM
             AND CA_NM      = CA.CA_NM
          )
       OR (USER_NM, CA_NM) IN
          (SELECT USER_NM, CA_NM
           FROM COL_CA
           WHERE (USER_NM_RT, RT_NM, USER_NM_CA, CA_NM) NOT IN
                   (SELECT USER_NM_RT, RT_NM, USER_NM_CA, CA_NM
                    FROM RT_CA
                    WHERE INS = 'YES'
                      AND DEL = 'NO'
                      AND RET = 'NO'
                   )
          )
      )
```

CA21- Tuples in COND_ACT for which CONSTR_TYPE = 'MVD' are referenced by at least one COL_CA-tuple for which GROUP_ID = 'DETERMINING', by at least one COL_CA-tuple with GROUP_ID = 'DEPENDENT', by at least one COL_CA-tuple with GROUP_ID = 'INDEPENDENT', by no other tuples in COL_CA, by one tuple in RT_CA, and by no tuple in any other R-table. The columns refer to the same R-table and the RT_CA-tuple has the value 'YES' values for the column INS 'No' values for the columns 'DEL' and 'RET'.

```
SELECT DISTINCT 'CA21: INCORRECT SPECIFICATION OF MULTI-VALUED DEPENDENCY CONSTRAINT'
FROM COND_ACT CA
WHERE CONSTR_TYPE = 'MVD'
  AND (
        (USER_NM, CA_NM) NOT IN
          (SELECT USER_NM, CA_NM
           FROM COL_CA
           WHERE GROUP_ID = 'DETERMINING'
          )
      OR (USER_NM, CA_NM) NOT IN
          (SELECT USER_NM, CA_NM
           FROM COL_CA
           WHERE GROUP_ID = 'DEPENDENT'
          )
      OR (USER_NM, CA_NM) NOT IN
          (SELECT USER_NM, CA_NM
           FROM COL_CA
           WHERE GROUP_ID = 'INDEPENDENT'
          )
      OR (USER_NM, CA_NM) IN
          (SELECT USER_NM, CA_NM
           FROM COL_CA
           WHERE GROUP_ID NOT IN ('DETERMINING','DEPENDENT','INDEPENDENT')
          )
      OR 1 <>
          (SELECT COUNT(*)
           FROM RT_CA
           WHERE USER_NM_CA = CA.USER_NM
             AND CA_NM      = CA.CA_NM
          )
      OR (USER_NM, CA_NM) IN
          (SELECT USER_NM, CA_NM
           FROM COL_CA
           WHERE (USER_NM_RT, RT_NM, USER_NM_CA, CA_NM) NOT IN
                   (SELECT USER_NM_RT, RT_NM, USER_NM_CA, CA_NM
                    FROM RT_CA
                    WHERE INS = 'YES'
                      AND DEL = 'NO'
                      AND RET = 'NO'
                   )
          )
      )
```

CA22- Tuples in COND_ACT for which CONSTR_TYPE = 'JD' are referenced by at least one COL_CA-tuple for which GROUP_ID = 'DETERMINING', by at least one COL_CA-tuple with a different GROUP_ID, by no other tuples in COL_CA, by one tuple in RT_CA, and by no tuple in any other R-table. The columns refer to the same R-table and the RT_CA-tuple has the value 'YES' values for the column INS 'No' values for the columns 'DEL' and 'RET'.

```
SELECT DISTINCT 'CA22: INCORRECT SPECIFICATION OF JOIN DEPENDENCY CONSTRAINT'
FROM COND_ACT CA
WHERE CONSTR_TYPE = 'JD'
  AND (   (USER_NM, CA_NM) NOT IN
          (SELECT USER_NM, CA_NM
           FROM COL_CA
           WHERE GROUP_ID = 'DETERMINING'
          )
       OR 3 >
          (SELECT COUNT(DISTINCT GROUP_ID)
           FROM COL_CA
           WHERE USER_NM_CA = CA.USER_NM
             AND CA_NM      = CA.CA_NM
          )
       OR 1 <>
          (SELECT COUNT(*)
           FROM RT_CA
           WHERE USER_NM_CA = CA.USER_NM
             AND CA_NM      = CA.CA_NM
          )
       OR (USER_NM, CA_NM) IN
          (SELECT USER_NM, CA_NM
           FROM COL_CA
           WHERE (USER_NM_RT, RT_NM, USER_NM_CA, CA_NM) NOT IN
                   (SELECT USER_NM_RT, RT_NM, USER_NM_CA, CA_NM
                    FROM RT_CA
                    WHERE INS = 'YES'
                      AND DEL = 'NO'
                      AND RET = 'NO'
                   )
          )
      )
```

CA23- **The set of COL_CA-tuples having the value 'DETERMINING' for the column GROUP_ID and referencing a COND_ACT-tuple for which CONSTR_TYPE = 'FD' does not include all primary key columns of the referenced R-table.**

```
SELECT DISTINCT 'CA23: FUNCTIONAL DEPENDENCY CONSTRAINT IS IMPLIED BY THE PRIMARY KEY'
FROM COND_ACT CA,
     KEY
WHERE CA.CONSTR_TYPE = 'FD'
  AND KEY.KEY_TYPE   = 'PRIMARY'
  AND NOT EXISTS
        (SELECT * FROM KEY_COLUMN KC
         WHERE KC.USER_NM = KEY.USER_NM
           AND KC.RT_NM   = KEY.RT_NM
           AND KC.KEY_NM  = KEY.KEY_NM
           AND NOT EXISTS
                 (SELECT * FROM COL_CA CCA
                  WHERE CCA.USER_NM_RT = KC.USER_NM
                    AND CCA.RT_NM      = KC.RT_NM
                    AND CCA.COL_NM     = KC.COL_NM
                    AND CCA.USER_NM_CA = CA.USER_NM
                    AND CCA.CA_NM      = CA.CA_NM
                    AND CCA.GROUP_ID   = 'DETERMINING'
                 )
        )
```

CA24- **Tuples in COND_ACT for which CONSTR_TYPE = 'INCL_DPCY' are referenced by pairs of COL_CA-tuples for which GROUP_ID = 'SUB' and for which GROUP_ID = 'SUPER', belonging to the same domain.**
This constraint cannot be fully expressed in SQL. The following statement does not return an error if the number of columns for which GROUP_ID = 'SUB' differs from the number of columns for which GROUP_ID = 'SUPER', but the set of domains to which the two groups of columns belong are identical.

```
SELECT DISTINCT 'CA24: INCORRECT SPECIFICATION OF INCLUSION DEPENDENCY CONSTRAINT'
FROM COND_ACT CA
WHERE CONSTR_TYPE = 'INCL_DPCY'
  AND (
        EXISTS
          (SELECT *
           FROM COL_CA  CCASUP,
                COLUMN  COLSUP
           WHERE CCASUP.USER_NM_CA = CA.USER_NM
             AND CCASUP.CA_NM      = CA.CA_NM
             AND CCASUP.USER_NM_RT = COLSUP.USER_NM
             AND CCASUP.RT_NM      = COLSUP.RT_NM
             AND CCASUP.COL_NM     = COLSUP.COL_NM
             AND CCASUP.GROUP_ID   = 'SUPER'
             AND NOT EXISTS
                   (SELECT *
                    FROM COL_CA  CCASUB,
                         COLUMN  COLSUB
                    WHERE CCASUB.USER_NM_CA = CA.USER_NM
                      AND CCASUB.CA_NM      = CA.CA_NM
                      AND CCASUB.USER_NM_RT = COLSUB.USER_NM
                      AND CCASUB.RT_NM      = COLSUB.RT_NM
                      AND CCASUB.COL_NM     = COLSUB.COL_NM
                      AND CCASUB.GROUP_ID   = 'SUB'
                      AND COLSUB.DOM_NM     = COLSUP.DOM_NM
                   )
          )
       OR
          EXISTS
            (SELECT *
             FROM COL_CA  CCASUB,
                  COLUMN  COLSUB
             WHERE CCASUB.USER_NM_CA = CA.USER_NM
               AND CCASUB.CA_NM      = CA.CA_NM
               AND CCASUB.USER_NM_RT = COLSUB.USER_NM
               AND CCASUB.RT_NM      = COLSUB.RT_NM
               AND CCASUB.COL_NM     = COLSUB.COL_NM
               AND CCASUB.GROUP_ID   = 'SUB'
               AND NOT EXISTS
                     (SELECT *
                      FROM COL_CA  CCASUP,
                           COLUMN  COLSUP
                      WHERE CCASUP.USER_NM_CA = CA.USER_NM
                        AND CCASUP.CA_NM      = CA.CA_NM
                        AND CCASUP.USER_NM_RT = COLSUP.USER_NM
                        AND CCASUP.RT_NM      = COLSUP.RT_NM
                        AND CCASUP.COL_NM     = COLSUP.COL_NM
                        AND CCASUP.GROUP_ID   = 'SUPER'
                        AND COLSUP.DOM_NM     = COLSUB.DOM_NM
                     )
            )
      )
```

CA25- Tuples in COL_CA, referring to a tuple in COND_ACT for which CONSTR_TYPE = 'INCL_DPCY' and having the same value for GROUP_ID, have the same values for RT_NM. The column GROUP_ID may only accept the values 'SUB' and 'SUPER'.

```
SELECT DISTINCT 'CA25: INCORRECT SPECIFICATION OF INCLUSION DEPENDENCY CONSTRAINT'
FROM COND_ACT CA
WHERE CONSTR_TYPE = 'INCL_DPCY'
  AND EXISTS
          (SELECT * FROM COL_CA CCA_1
           WHERE CCA_1.USER_NM_CA = CA.USER_NM
             AND CCA_1.CA_NM     = CA.CA_NM
             AND (   CCA_1.GROUP_ID NOT IN ('SUB','SUPER')
                  OR EXISTS
                        (SELECT * FROM COL_CA CCA_2
                         WHERE CCA_2.USER_NM_CA = CCA_1.USER_NM_CA
                           AND CCA_2.CA_NM      = CCA_1.CA_NM
                           AND CCA_2.GROUP_ID   = CCA_1.GROUP_ID
                           AND CCA_2.RT_NM      <> CCA_1.RT_NM
                        )
                 )
          )
```

CA26- Columns in RT_CA referencing a COND_ACT-tuple for which CONSTR_TYPE = 'INCL_DPCY' and referenced by COL_CA-tuples for which GROUP_ID = 'SUPER' have the value 'YES' for the column DEL and the value 'NO' for their columns INS and RET, while columns in RT_CA referencing a COND_ACT-tuple for which CONSTR_TYPE = 'INCL_DPCY' and referenced by COL_CA-tuples for which GROUP_ID = 'SUB' have the value 'YES' for the column INS and the value 'NO' for their columns DEL and RET.

```
SELECT DISTINCT 'CA26: INCORRECT SPECIFICATION OF INCLUSION DEPENDENCY CONSTRAINT'
FROM COND_ACT CA
WHERE CONSTR_TYPE = 'INCL_DPCY'
  AND (
        NOT EXISTS
          (SELECT * FROM RT_CA RCA
           WHERE RCA.USER_NM_CA = CA.USER_NM
             AND RCA.CA_NM      = CA.CA_NM
             AND INS = 'NO'
             AND DEL = 'YES'
             AND RET = 'NO'
             AND EXISTS
                   (SELECT * FROM COL_CA CCA
                    WHERE CCA.USER_NM_CA = RCA.USER_NM_CA
                      AND CCA.CA_NM      = RCA.CA_NM
                      AND CCA.USER_NM_RT = RCA.USER_NM_RT
                      AND CCA.RT_NM      = RCA.RT_NM
                      AND CCA.GROUP_ID   = 'SUPER'
                   )
          )
      OR
        NOT EXISTS
          (SELECT * FROM RT_CA RCA
           WHERE RCA.USER_NM_CA = CA.USER_NM
             AND RCA.CA_NM      = CA.CA_NM
             AND INS = 'YES'
             AND DEL = 'NO'
             AND RET = 'NO'
             AND EXISTS
                   (SELECT * FROM COL_CA CCA
                    WHERE CCA.USER_NM_CA = RCA.USER_NM_CA
                      AND CCA.CA_NM      = RCA.CA_NM
                      AND CCA.USER_NM_RT = RCA.USER_NM_RT
                      AND CCA.RT_NM      = RCA.RT_NM
                      AND CCA.GROUP_ID   = 'SUB'
                   )
          )
      )
```

CA27- The set of tuples in COL_CA, referring to a tuple in COND_ACT for which CONSTR_TYPE = 'INCL_DPCY', and for which GROUP_ID = 'SUB', does not coincide with a set of KEY_COLUMN-tuples referring to a KEY-tuple for which KEY_TYPE = 'FOREIGN', while at the same time the corresponding set of tuples in COL_CA, for which GROUP_ID = 'SUPER', refers to a KEY-tuple for which KEY_TYPE = 'PRIMARY'.

```
SELECT DISTINCT 'CA27: INCLUSION DEPENDENCY IS IMPLIED BY REFERENTIAL INTEGRITY CONSTRAINT'
FROM COND_ACT CA,
     KEY      PK,
     KEY      FK,
     KEY_REF  KR
WHERE CA.CONSTR_TYPE = 'INCL_DPCY'
  AND PK.KEY_TYPE    = 'PRIMARY'
  AND FK.KEY_TYPE    = 'FOREIGN'
  AND KR.USER_NM_PK  = PK.USER_NM
  AND KR.RT_NM_PK    = PK.RT_NM
  AND KR.USER_NM_FK  = FK.USER_NM
  AND KR.RT_NM_FK    = FK.RT_NM
  AND KR.KEY_NM      = FK.KEY_NM
  AND NOT EXISTS  /* every super-column is part of the primary key */
        (SELECT * FROM COL_CA CCA
         WHERE CCA.USER_NM_CA = CA.USER_NM
           AND CCA.CA_NM      = CA.CA_NM
           AND CCA.GROUP_ID   = 'SUPER'
           AND NOT EXISTS
                 (SELECT * FROM KEY_COLUMN KC
                  WHERE KC.USER_NM = PK.USER_NM
                    AND KC.RT_NM   = PK.RT_NM
                    AND KC.KEY_NM  = PK.KEY_NM
                    AND KC.USER_NM = CCA.USER_NM_RT
                    AND KC.RT_NM   = CCA.RT_NM
                    AND KC.COL_NM  = CCA.COL_NM
                 )
        )
  AND NOT EXISTS  /* every primary-key column is exists as a super-column */
        (SELECT * FROM KEY_COLUMN KC
         WHERE KC.USER_NM = PK.USER_NM
           AND KC.RT_NM   = PK.RT_NM
           AND KC.KEY_NM  = PK.KEY_NM
           AND NOT EXISTS
                 (SELECT * FROM COL_CA CCA
                  WHERE CCA.USER_NM_CA = CA.USER_NM
                    AND CCA.CA_NM      = CA.CA_NM
                    AND CCA.USER_NM_RT = KC.USER_NM
                    AND CCA.RT_NM      = KC.RT_NM
                    AND CCA.COL_NM     = KC.COL_NM
                    AND CCA.GROUP_ID   = 'SUPER'
                 )
        )
  AND NOT EXISTS  /* every sub-column is part of the foreign key */
        (SELECT * FROM COL_CA CCA
         WHERE CCA.USER_NM_CA = CA.USER_NM
           AND CCA.CA_NM      = CA.CA_NM
           AND CCA.GROUP_ID   = 'SUB'
           AND NOT EXISTS
                 (SELECT * FROM KEY_COLUMN KC
                  WHERE KC.USER_NM = FK.USER_NM
                    AND KC.RT_NM   = FK.RT_NM
                    AND KC.KEY_NM  = FK.KEY_NM
                    AND KC.USER_NM = CCA.USER_NM_RT
                    AND KC.RT_NM   = CCA.RT_NM
                    AND KC.COL_NM  = CCA.COL_NM
                 )
        )
  AND NOT EXISTS  /* every foreign-key column exists as a sub-column */
        (SELECT * FROM KEY_COLUMN KC
         WHERE KC.USER_NM = FK.USER_NM
           AND KC.RT_NM   = FK.RT_NM
           AND KC.KEY_NM  = FK.KEY_NM
           AND NOT EXISTS
                 (SELECT * FROM COL_CA CCA
                  WHERE CCA.USER_NM_CA = CA.USER_NM
                    AND CCA.CA_NM      = CA.CA_NM
                    AND CCA.USER_NM_RT = KC.USER_NM
                    AND CCA.RT_NM      = KC.RT_NM
                    AND CCA.COL_NM     = KC.COL_NM
                    AND CCA.GROUP_ID   = 'SUB'
                 )
        )
```

CA28- For every DOMAIN-tuple the columns USER_NM_CA and CA_NM are either I-marked or not marked.

```
SELECT DISTINCT 'CA28: USER_NM_CA AND CA_NM IN DOMAIN ARE PARTLY A-MARKED'
FROM DOMAIN
WHERE (    USER_NM_CA IS NULL       /*     a-marked */
       AND CA_NM IS NOT NULL        /* not a-marked */
      )
   OR (    USER_NM_CA IS NOT NULL   /*     a-marked */
       AND CA_NM IS NULL            /* not a-marked */
      )
```

8. BUILT-IN AND USER-DEFINED FUNCTIONS

FU01- If FUNC_NM_INV is not A-marked, the FUNCTION-tuple it references contains the value of FUNC_NM for its column FUNC_NM_INV.

```
SELECT DISTINCT 'FU01: INCONSISTENT INVERSE FUNCTION REFERENCES'
FROM FUNCTION F1
WHERE FUNC_NM_INV IS NOT NULL  /* not a-marked */
  AND (USER_NM, FUNC_NM_INV) NOT IN
        (SELECT USER_NM, FUNC_NM
         FROM FUNCTION F2
         WHERE F2.FUNC_NM_INV = F1.FUNC_NM
        )
```

FU02- If FUNC_NM_DOD is not A-marked, the FUNCTION-tuple it references contains the value of FUNC_NM for its column FUNC_NM_DOD.

```
SELECT DISTINCT 'FU02: INCONSISTENT DOD-FUNCTION REFERENCES'
FROM FUNCTION F1
WHERE FUNC_NM_DOD IS NOT NULL  /* not a-marked */
  AND (USER_NM, FUNC_NM_DOD) NOT IN
        (SELECT USER_NM, FUNC_NM
         FROM FUNCTION F2
         WHERE F2.FUNC_NM_DOD = F1.FUNC_NM
        )
```

FU03- If FUNC_NM_DOD is not A-marked, then IND_DOD must not be I-marked.

```
SELECT DISTINCT 'FU03: REFERENCE TO DOD-FUNCTION EXISTS AND IND_DOD IS I-MARKED'
FROM FUNCTION
WHERE FUNC_NM_DOD IS NOT NULL  /* not a-marked */
  AND IND_DOD     IS NULL      /*     i-marked */
```

FU04- Functions for which FUNC_NM_DOD is not I-marked reference FUNCTION-tuples for which IND_DOD has a different value.

```
SELECT DISTINCT 'FU04: REFERENCED DOD-FUNCTION DOES NOT HAVE A DIFFERENT VALUE FOR IND_DOD'
FROM FUNCTION F1
WHERE FUNC_NM_DOD IS NOT NULL  /* not a-marked */
  AND (USER_NM, FUNC_NM_DOD) IN
        (SELECT USER_NM, FUNC_NM
         FROM FUNCTION F2
         WHERE F1.IND_DOD = F2.IND_DOD
        )
```

FU05- R_TABLE-tuples for which RT_TYPE <> 'VIEW' or INS_ALL = 'NO' have A-marks for FUNC_NM_VW.

```
SELECT DISTINCT 'FU05: REFERENCE TO INSERT-FUNCTION ONLY ALLOWED FOR INSERTABLE VIEWS'
FROM R_TABLE
WHERE FUNC_NM_VW IS NOT NULL  /* not a-marked */
  AND (    INS_ALL = 'NO'
       OR RT_TYPE <> 'VIEW'
      )
```

FU06- For every tuple in R_TABLE for which FUNC_NM_VW is not A-marked there exists a tuple in RT_FUNC with the same values for RT_NM and FUNC_NM and with values for USER_NM_RT and USER_NM_FUNC that are equal to USER_NM in R_TABLE.

```
SELECT DISTINCT 'FU06: REFERENCE TO INSERT-FUNCTION REQUIRES RT_FUNC DATA'
FROM R_TABLE
WHERE FUNC_NM_VW IS NOT NULL  /* not a-marked */
  AND (USER_NM, RT_NM, USER_NM, FUNC_NM_VW) NOT IN
        (SELECT USER_NM_RT, RT_NM, USER_NM_FUNC, FUNC_NM
         FROM RT_FUNC
        )
```

FU07- **FUNCTION-tuples that are each others inverse or that constitute a DOD/non-DOD pair are referenced by identical sets of ARGUMENT-tuples, projected on columns ARG_SEQ# and ARG_NM.**

```
SELECT DISTINCT 'FU07: ARGUMENT LISTS OF INVERSE FUNCTIONS DO NOT MATCH'
FROM FUNCTION F1, FUNCTION F2
WHERE (
        (    F1.USER_NM = F2.USER_NM
         AND F1.FUNC_NM = F2.FUNC_NM_INV   /* inverse function pair */
        )
       OR
        (    F1.USER_NM = F2.USER_NM
         AND F1.FUNC_NM = F2.FUNC_NM_DOD   /* dod/non-dod function pair */
        )
      )
  AND EXISTS
      (SELECT * FROM ARGUMENT ARG1
       WHERE ARG1.USER_NM = F1.USER_NM
         AND ARG1.FUNC_NM = F1.FUNC_NM
         AND NOT EXISTS
             (SELECT * FROM ARGUMENT ARG2
              WHERE ARG2.USER_NM  = F2.USER_NM
                AND ARG2.FUNC_NM  = F2.FUNC_NM
                AND ARG2.ARG_SEQ# = ARG1.ARG_SEQ#
                AND ARG2.ARG_NM   = ARG1.ARG_NM
             )
      )
```

FU08- **The values of ARG_SEQ# in the set of ARGUMENT-tuples with the same value for USER_NM and FUNC_NM are numbered consecutively up from 1 to the number of tuples in the set.**

```
SELECT DISTINCT 'FU08: ARGUMENTS MUST BE NUMBERED CONSECUTIVELY'
FROM ARGUMENT ARG1
WHERE ARG1.ARG_SEQ# > 1
  AND NOT EXISTS
     (SELECT * FROM ARGUMENT ARG2
      WHERE ARG1.USER_NM  = ARG2.USER_NM
        AND ARG1.ARG_SEQ# = ARG2.ARG_SEQ# + 1
     )
```

FU09- **The relation SINGULAR contains exactly one tuple.**

```
SELECT DISTINCT 'FU09: R-TABLE SINGULAR MUST CONSTAIN EXACTLY ONE TUPLE'
FROM DUAL  /* oracle's dummy table */
WHERE 1 <>
       (SELECT COUNT(*)
        FROM SINGULAR
       )
```

FU10- **The values of columns DOM_NM_FUNC and DOM_NM_ARG in SINGULAR are not equal.**

```
SELECT DISTINCT 'FU10: FUNCTION DOMAIN AND ARGUMENT DOMAIN MUST BE DIFFERENT'
FROM SINGULAR
WHERE DOM_NM_FUNC = DOM_NM_ARG
```

FU11- **If the USER_NM in FUNCTION is not equal to USER_NM_CATALOG in SINGULAR, the columns HOST_LANG, ROUT_SRC and ROUT_EXEC must not be marked.**

```
SELECT DISTINCT 'FU11: HOST LANGUAGE SOURCE ROUTINE AND EXECUTABLE ROUTINE MUST BE KNOWN'
FROM FUNCTION
WHERE (   HOST_LANG IS NULL  /* i-marked */
       OR ROUT_SRC  IS NULL  /* i-marked */
       OR ROUT_EXEC IS NULL  /* i-marked */
      )
  AND USER_NM <>
     (SELECT USER_NM_CATALOG
      FROM SINGULAR
     )
```

FU12- **Every DOMAIN-tuple that is referenced from SINGULAR has values for USER_NM_DOM and USER_NM_CA that are equal to USER_NM_CATALOG in SINGULAR.**

```
SELECT DISTINCT 'FU12: BUILT-IN DOMAIN MUST BE OWNED BY THE CATALOG'
FROM DOMAIN D,
     SINGULAR S
WHERE (   D.DOM_NM = S.DOM_NM_FUNC
       OR D.DOM_NM = S.DOM_NM_ARG
      )
  AND (   D.USER_NM_DOM <> S.USER_NM_CATALOG
       OR D.USER_NM_CA  <> S.USER_NM_CATALOG
       OR D.USER_NM_DOM IS NULL   /* a-marked */
      )
```

9. SYNONYM NAMES AND NAMING RULES

NR01- **The sets of R_TABLE-tuples designated by columns USER_NM_SYN and RT_NM_SYN on the one hand and by USER_NM_RT and RT_NM in RT_SYN are disjunct.**

```
SELECT DISTINCT 'NR01: SYNONYM NAME CANNOT COINCIDE WITH AN R-TABLE NAME'
FROM RT_SYN
WHERE (USER_NM_SYN, RT_NM_SYN) IN
        (SELECT USER_NM_RT, RT_NM
         FROM RT_SYN)
```

NR02- **The set of composite column names designated by COL_NM_SUP in COL_STRUCT is disjunct from the set of keywords in the relation RL_KEYWORD.**

A DOMAIN CHECK OVERRIDE is required here.

```
SELECT DISTINCT 'NR02: COMPOSITE COLUMN NAME CANNOT COINCIDE WITH AN RL-KEYWORD'
FROM COLUMN_STRUCT
WHERE COL_NM_SUP IN
        (SELECT RL_KEYW
         FROM RL_KEYWORD
        )
```

NR03- **The set of domain names designated by DOM_NM in DOMAIN is disjunct from the union of RT_NM_SYN in RT_SYN and RT_NM in R_TABLE.**

A DOMAIN CHECK OVERRIDE is required here.

```
SELECT DISTINCT 'NR03: DOMAIN NAME CANNOT COINCIDE WITH AN R-TABLE NAME OR A FUNCTION NAME'
FROM DOMAIN
WHERE DOM_NM IN
        (SELECT RT_NM_SYN
         FROM RT_SYN
        )
   OR DOM_NM IN
        (SELECT RT_NM
         FROM R_TABLE
        )
   OR DOM_NM IN
        (SELECT FUNC_NM
         FROM FUNCTION
        )
```

NR04- For every user, the sets of names designated by RT_NM_SYN in RT_SYN, RT_NM in R_TABLE and FUNC_NM in FUNCTION are distinct from another and from the union of DOM_NM in DOMAIN, BD_TYPE_NM in BASIC_DATATYPE, and COL_NM in COLUMN for that user.
A DOMAIN CHECK OVERRIDE is required here.

```
SELECT DISTINCT 'NR04: NAMES OF R-TABLES, FUNCTIONS, COLUMNS, DOMAINS AND BASIC DATATYPES MUST
NOT COINCIDE'
FROM DUAL  /* oracle's dummy table */
WHERE EXISTS
        (SELECT * FROM FUNCTION
         WHERE (USER_NM, FUNC_NM) IN
                 (SELECT USER_NM_SYN, RT_NM_SYN
                  FROM RT_SYN
                 )
        )
    OR EXISTS
        (SELECT * FROM FUNCTION
         WHERE (USER_NM, FUNC_NM) IN
                 (SELECT USER_NM, RT_NM
                  FROM R_TABLE
                 )
        )
    OR EXISTS
        (SELECT * FROM FUNCTION
         WHERE (USER_NM, FUNC_NM) IN
                 (SELECT USER_NM, COL_NM
                  FROM COLUMN
                 )
        )
    OR EXISTS
        (SELECT * FROM RT_SYN
         WHERE (USER_NM_SYN, RT_NM_SYN) IN
                 (SELECT USER_NM, COL_NM
                  FROM COLUMN
                 )
        )
    OR EXISTS
        (SELECT * FROM R_TABLE
         WHERE (USER_NM, RT_NM) IN
                 (SELECT USER_NM, COL_NM
                  FROM COLUMN
                 )
        )
    OR EXISTS
        (SELECT * FROM FUNCTION
         WHERE FUNC_NM IN
                 (SELECT DOM_NM
                  FROM DOMAIN
                  UNION
                  SELECT BD_TYPE_NM
                  FROM BASIC_DATATYPE
                 )
        )
    OR EXISTS
        (SELECT * FROM R_TABLE
         WHERE RT_NM IN
                 (SELECT DOM_NM
                  FROM DOMAIN
                  UNION
                  SELECT BD_TYPE_NM
                  FROM BASIC_DATATYPE
                 )
        )
    OR EXISTS
        (SELECT * FROM RT_SYN
         WHERE RT_NM_SYN IN
                 (SELECT DOM_NM
                  FROM DOMAIN
                  UNION
                  SELECT BD_TYPE_NM
                  FROM BASIC_DATATYPE
                 )
        )
```

NR05- Every DOMAIN-tuple and every COLUMN-tuple, that references a BASIC_DATATYPE-tuple for which IND_LENGTH = 'YES', does not have a marked value for its LENGTH column, while the LENGTH column is I-marked if IND_LENGTH = 'NO'.

```
SELECT DISTINCT 'NR05: LENGTH MUST BE SPECIFIED OR MUST NOT BE SPECIFIED'
FROM BASIC_DATATYPE
WHERE (    IND_LENGTH = 'YES'
       AND BD_TYPE_NM IN
                (SELECT BD_TYPE_NM
                 FROM COLUMN
                 WHERE LENGTH IS NULL  /* i-marked */
                )
      )
   OR (    IND_LENGTH = 'YES'
       AND BD_TYPE_NM IN
                (SELECT BD_TYPE_NM
                 FROM DOMAIN
                 WHERE LENGTH IS NULL  /* i-marked */
                )
      )
   OR (    IND_LENGTH = 'NO'
       AND BD_TYPE_NM IN
                (SELECT BD_TYPE_NM
                 FROM COLUMN
                 WHERE LENGTH IS NOT NULL  /* not i-marked */
                )
      )
   OR (    IND_LENGTH = 'NO'
       AND BD_TYPE_NM IN
                (SELECT BD_TYPE_NM
                 FROM DOMAIN
                 WHERE LENGTH IS NOT NULL  /* not i-marked */
                )
      )
```

NR06- Every DOMAIN-tuple and every COLUMN-tuple, that references a BASIC_DATATYPE-tuple for which IND_SCALE = 'YES', does not have a marked value for its SCALE column, while the SCALE column is I-marked if IND_SCALE = 'NO'.

```
SELECT DISTINCT 'NR06: SCALE MUST BE SPECIFIED OR MUST NOT BE SPECIFIED'
FROM BASIC_DATATYPE
WHERE (    IND_SCALE = 'YES'
       AND BD_TYPE_NM IN
                (SELECT BD_TYPE_NM
                 FROM COLUMN
                 WHERE SCALE IS NULL  /* i-marked */
                )
      )
   OR (    IND_SCALE = 'YES'
       AND BD_TYPE_NM IN
                (SELECT BD_TYPE_NM
                 FROM DOMAIN
                 WHERE SCALE IS NULL  /* i-marked */
                )
      )
   OR (    IND_SCALE = 'NO'
       AND BD_TYPE_NM IN
                (SELECT BD_TYPE_NM
                 FROM COLUMN
                 WHERE SCALE IS NOT NULL  /* not i-marked */
                )
      )
   OR (    IND_SCALE = 'NO'
       AND BD_TYPE_NM IN
                (SELECT BD_TYPE_NM
                 FROM DOMAIN
                 WHERE SCALE IS NOT NULL  /* not i-marked */
                )
      )
```

11. AUTHORIZATION

AU01- **The values of the columns N_DROP, N_DELETE and N_DELAY in RT_SITE must be less than the number of individual users in the relation USER_SITE.**

```
SELECT DISTINCT 'AU01: N-PERSON TURN KEY CANNOT EXCEED THE NUMBER OF DATABASE USERS'
FROM RT_SITE
WHERE N_DROP >=
        (SELECT COUNT(*)
         FROM USER_SITE
         WHERE SITE_NM = RT_SITE.SITE_NM
           AND USER_NM NOT IN
                (SELECT USER_GRP_NM
                 FROM USER
                 WHERE USER_GRP_NM IS NOT NULL
                )
        )
   OR N_DELETE >=
        (SELECT COUNT(*)
         FROM USER_SITE
         WHERE SITE_NM = RT_SITE.SITE_NM
           AND USER_NM NOT IN
                (SELECT USER_GRP_NM
                 FROM USER
                 WHERE USER_GRP_NM IS NOT NULL
                )
        )
   OR N_DELAY >=
        (SELECT COUNT(*)
         FROM USER_SITE
         WHERE SITE_NM = RT_SITE.SITE_NM
           AND USER_NM NOT IN
                (SELECT USER_GRP_NM
                 FROM USER
                 WHERE USER_GRP_NM IS NOT NULL
                )
        )
```

AU02- **A COLUMN-tuple contains an I-MARK for MARK_PREF if and only if A_MARK_ALL and I_MARK_ALL both contain the value 'YES'.**

```
SELECT DISTINCT 'AU02: MARK-PREFERENCE REQUIRED OR NOT RELEVANT'
FROM COLUMN
WHERE (    MARK_PREF IS NOT NULL  /* not i-marked */
       AND (   A_MARK_ALL = 'NO'
            OR I_MARK_ALL = 'NO'
           )
      )
   OR (    MARK_PREF IS NULL  /* i-marked */
       AND A_MARK_ALL = 'YES'
       AND I_MARK_ALL = 'YES'
      )
```

AU03- **No RT_AUTH-tuple has the same values for USER_NM_RT and USER_NM_GRANTEE or USER_NM_GRANTOR and USER_NM_GRANTEE.**

```
SELECT DISTINCT 'AU03: A USER CANNOT GRANT AUTHORIZATION TO HIMSELF OR HERSELF'
FROM RT_AUTH
WHERE USER_NM_RT      = USER_NM_GRANTEE
   OR USER_NM_GRANTOR = USER_NM_GRANTEE
```

AU04- **Authorization cycles may not occur in RT_AUTH.**
This constraint cannot be expressed in RL.

AU05- **For every RT_AUTH-tuple, a column containing an RL-expression must contain an I-mark if the corresponding indicator contains the value 'NO'.**

```
SELECT 'AU05: RL-EXPRESION NOT I-MARKED AND INDICATOR = "NO"'
FROM RT_AUTH
WHERE (    RET_ALL = 'NO'
       AND RL_EXPR_RET IS NOT NULL
      )
   OR (    INS_ALL = 'NO'
       AND RL_EXPR_INS IS NOT NULL
      )
   OR (    PK_UPD_ALL = 'NO'
       AND RL_EXPR_PK_UPD IS NOT NULL
      )
   OR (    ARCH_ALL = 'NO'
       AND RL_EXPR_ARCH IS NOT NULL
      )
   OR (    DEL_ALL = 'NO'
       AND RL_EXPR_DEL IS NOT NULL
      )
   OR (    SNAPSHOT_ALL = 'NO'
       AND RL_EXPR_SNAPSHOT IS NOT NULL
      )
   OR (    CHECKOUT_ALL = 'NO'
       AND RL_EXPR_CHECKOUT IS NOT NULL
      )
```

AU06- Every RT_AUTH-tuple for which RET_ALL = 'NO' has the same value for SNAPSHOT_ALL.

```
SELECT DISTINCT 'AU06: AUTHORIZATION FOR SHAPSHOT CREATION IS MEANINGLESS'
FROM RT_AUTH
WHERE RET_ALL = 'NO'
  AND SNAPSHOT_ALL = 'YES'
```

AU07- For every COL_AUTH-tuple, the column RL_EXPR_UPD must contain an I-mark if the corresponding indicator UPD_ALL contains the value 'NO'.

```
SELECT 'AU07: RL-EXPRESION NOT I-MARKED AND INDICATOR = "NO"'
FROM COL_AUTH
WHERE UPD_ALL = 'NO'
  AND RL_EXPR_UPD IS NOT NULL
```

AU08- The column DEFAULT_VAL in COL_AUTH must conform to the domain specification of the domain to which the referenced COLUMN-tuple belongs.
This constraint cannot be expressed in RL.

AU09- The column DEFAULT_VAL in TERM_DFLT must conform to the domain specification of the domain to which the referenced COLUMN-tuple belongs.
This constraint cannot be expressed in RL.

AU10- The column DEFAULT_VAL in PROG_AUTH must conform to the domain specification of the domain to which the referenced COLUMN-tuple belongs.
This constraint cannot be expressed in RL.

AU11- Every USER-tuple that is referenced by another USER-tuple has an A-mark for USER_GRP_NM.

```
SELECT DISTINCT 'AU11: MULTI-LEVEL USER-GROUP HIERARCHY NOT ALLOWED'
FROM USER
WHERE USER_GRP_NM IS NOT NULL   /* a-marked */
  AND USER_NM IN
       (SELECT USER_GRP_NM
        FROM USER
        WHERE USER_GRP_NM IS NOT NULL   /* a-marked */
       )
```

AU12- For no USER_SITE-tuple the storage authorizations per query (MAX_STOR_QUERY_ANT, MAX_STOR_QUERY_ACT) exceed their total counterparts (MAX_STOR_TOTAL_ANT, MAX_STOR_TOTAL_ACT).

```
SELECT DISTINCT 'AU12: RESOURCE USAGE ALLOWANCE PER QUERY EXCEEDS TOTAL ALLOWANCE'
FROM USER_SITE
WHERE MAX_STOR_QUERY_ANT > MAX_STOR_TOTAL_ANT
   OR MAX_STOR_QUERY_ACT > MAX_STOR_TOTAL_ACT
```

AU13- The column PK_HIDDEN in RT_AUTH must not contain an I-mark if and only if the referenced R-table has a primary key.

```
SELECT DISTINCT 'AU13: COLUMN PK_HIDDEN MUST (NOT) BE SPECIFIED'
FROM RT_AUTH
WHERE (    PK_HIDDEN IS NULL   /* i-marked */
       AND (USER_NM_RT, RT_NM) IN
              (SELECT USER_NM, RT_NM
               FROM KEY
               WHERE KEY_TYPE = 'PRIMARY'
              )
      )
   OR (    PK_HIDDEN IS NOT NULL   /* not i-marked */
       AND (USER_NM_RT, RT_NM) NOT IN
              (SELECT USER_NM, RT_NM
               FROM KEY
               WHERE KEY_TYPE = 'PRIMARY'
              )
      )
```

AU14- The column CHECKOUT_ALL in RT_AUTH must not contain an I-mark if and only if the referenced R-table is a base R-table.

```
SELECT DISTINCT 'AU14: AUTHORIZING FOR CHECHOUT IS MANDATORY OR IRRELEVANT'
FROM RT_AUTH
WHERE (    CHECKOUT_ALL IS NULL   /* i-marked */
       AND (USER_NM_RT, RT_NM) IN
              (SELECT USER_NM, RT_NM
               FROM R_TABLE
               WHERE RT_TYPE = 'BASE'
              )
      )
   OR (    CHECKOUT_ALL IS NOT NULL   /* i-marked */
       AND (USER_NM_RT, RT_NM) NOT IN
              (SELECT USER_NM, RT_NM
               FROM R_TABLE
               WHERE RT_TYPE = 'BASE'
              )
      )
```

AU15- Every USER_SITE-tuple for which IND_DBA_USER = 'YES' (local DBA) has the value 'YES' for GRANT_ALL.

```
SELECT DISTINCT 'AU15: DBA-USER MUST HAVE AUTHORIZATIONS FOR GRANTING'
FROM USER_SITE
WHERE IND_DBA_USER = 'YES'
  AND GRANT_ALL    = 'NO'
```

12. AUDIT LOGGING

LG01- **Every tuple in AUDIT_LOG references a tuple in COLUMN that references a tuple in R_TABLE for which RT_TYPE = 'BASE'**

```
SELECT DISTINCT 'LG01: LOGGING CAN ONLY TAKE PLACE ON BASE R-TABLES'
FROM AUDIT_LOG
WHERE (USER_NM_RT, RT_NM) NOT IN
        (SELECT USER_NM, RT_NM
         FROM R_TABLE
         WHERE RT_TYPE = 'BASE'
        )
```

LG02- **Every tuple in AUDIT_LOG is referenced by at least one tuple in LOG_PK_COLUMN.**

```
SELECT DISTINCT 'LG02: AUDIT_LOG-TUPLE NOT REFERENCED BY LOG_PK_COLUMN-TUPLE'
FROM AUDIT_LOG
WHERE AUDIT_SEQ# NOT IN
        (SELECT AUDIT_SEQ# FROM LOG_PK_COLUMN)
```

LG03- **The set of COLUMN-tuples referenced by tuples in LOG_PK_COLUMN and AUDIT_LOG, for which AUDIT_SEQ# is the same, all reference the same R_TABLE-tuple.**

```
SELECT DISTINCT 'LG03: LOGGED PRIMARY KEY COLUMN DOES NOT BELONG TO THE CORRECT R-TABLE'
FROM LOG_PK_COLUMN LPK
WHERE EXISTS
        (SELECT * FROM AUDIT_LOG LOG
         WHERE LPK.USER_NM <> LOG.USER_NM_RT
            OR LPK.RT_NM   <> LOG.RT_NM
)
```

LG04- **For every tuple in AUDIT_INTERVAL the columns DATE_LOG_OFF, TIME_LOG_OFF and USER_NM_LOG_OFF are all either marked or unmarked.**

```
SELECT DISTINCT 'LG04: LOG-OFF DATA CANNOT BE PARTLY MARKED'
FROM AUDIT_INTERVAL
WHERE (   DATE_LOG_OFF IS NULL                /*     a-marked */
      AND (   TIME_LOG_OFF IS NOT NULL        /* not a-marked */
           OR USER_NM_LOG_OFF IS NOT NULL     /* not a-marked */
          )
      )
   OR (   TIME_LOG_OFF IS NULL                /*     a-marked */
      AND (   DATE_LOG_OFF IS NOT NULL        /* not a-marked */
           OR USER_NM_LOG_OFF IS NOT NULL     /* not a-marked */
          )
      )
   OR (   USER_NM_LOG_OFF IS NULL             /*     a-marked */
      AND (   DATE_LOG_OFF IS NOT NULL        /* not a-marked */
           OR TIME_LOG_OFF IS NOT NULL        /* not a-marked */
          )
      )
```

13. DISTRIBUTED DATABASE

DD01- **Every tuple in R_TABLE is referenced by at least one RT_SITE-tuple.**

```
SELECT DISTINCT 'DD01: R_TABLE-TUPLE NOT REFERENCED BY RT_SITE-TUPLE'
FROM R_TABLE
WHERE (USER_NM, RT_NM) NOT IN
        (SELECT USER_NM, RT_NM FROM RT_SITE)
```

DD02- **Every tuple in RT_SITE is referenced by at least one COL_SITE-tuple.**

```
SELECT DISTINCT 'DD02: RT_SITE-TUPLE NOT REFERENCED BY COL_SITE-TUPLE'
FROM RT_SITE
WHERE (USER_NM, RT_NM) NOT IN
        (SELECT USER_NM, RT_NM FROM COL_SITE)
```

DD03- The RT_SITE-columns RL_EXPR_LG, RL_EXPR_GL and IND_REPLICA must not contain I-marks if and only if the column RT_TYPE in R_TABLE = 'BASE'.

```
SELECT DISTINCT 'DD03: COLUMNS RL_EXPR_LG, RL_EXPR_GL AND IND_REPLICA MUST (NOT) BE MARKED'
FROM RT_SITE
WHERE (
        (USER_NM, RT_NM) IN
          (SELECT USER_NM, RT_NM
           FROM R_TABLE
           WHERE RT_TYPE = 'BASE'
          )
    AND (   RL_EXPR_LG  IS NULL /* i-marked */
         OR RL_EXPR_GL  IS NULL /* i-marked */
         OR IND_REPLICA IS NULL /* i-marked */
        )
      )
  OR (
        (USER_NM, RT_NM) NOT IN
          (SELECT USER_NM, RT_NM
           FROM R_TABLE
           WHERE RT_TYPE = 'BASE'
          )
    AND (   RL_EXPR_LG  IS NOT NULL /* not i-marked */
         OR RL_EXPR_GL  IS NOT NULL /* not i-marked */
         OR IND_REPLICA IS NOT NULL /* not i-marked */
        )
     )
```

DD04- There exists at least one SITE-tuple for which IND_CATALOG_SITE equals 'YES'.

```
SELECT DISTINCT 'DD04: CATALOG SITE UNKNOWN'
FROM DUAL    /* oracle's dummy table */
WHERE 'YES' NOT IN
       (SELECT IND_CATALOG_SITE FROM SITE)
```

DD05- Every tuple in USER is referenced by at least one USER_SITE-tuple.

```
SELECT DISTINCT 'DD05: USER-TUPLE NOT REFERENCED BY USER_SITE-TUPLE'
FROM USER
WHERE USER_NM NOT IN
       (SELECT USER_NM FROM USER_SITE)
```

DD06- Every tuple in SITE is referenced by at least one USER_SITE-tuple.

```
SELECT DISTINCT 'DD06: SITE-TUPLE NOT REFERENCED BY USER_SITE-TUPLE'
FROM SITE
WHERE SITE_NM NOT IN
       (SELECT SITE_NM FROM USER_SITE)
```

DD07- For every individual user for which IND_GDBA = 'YES' the number of USER_SITE-tuples referencing it equals the number of SITE-tuples.

```
SELECT DISTINCT 'DD07: GLOBAL DBA USER MUST BE KNOWN ASSIGNED TO EVERY SITE'
FROM USER
WHERE IND_GDBA = 'YES'
  AND EXISTS
       (SELECT * FROM SITE
        WHERE NOT EXISTS
               (SELECT * FROM USER_SITE
                WHERE USER_NM = USER.USER_NM
                  AND SITE_NM = SITE.SITE_NM
               )
       )
  AND USER_NM NOT IN
       (SELECT USER_GRP_NM
        FROM USER
        WHERE USER_GRP_NM IS NOT NULL
       )
```

DD08- The USER-tuple referenced by the SINGULAR-tuple has the value 'YES' for IND_GDBA.

```
SELECT DISTINCT 'DD08: CATALOG OWNER MUST BE A GLOBAL DBA'
FROM USER
WHERE USER_NM  = (SELECT USER_NM_CATALOG FROM SINGULAR)
  AND IND_GDBA = 'NO'
```

DD09- Every USER_SITE-tuple that references a USER-tuple for which IND_GDBA = 'YES' has the value 'YES' for IND_DBA_USER.

```
SELECT DISTINCT 'DD09: GLOBAL DBA USER MUST BE KNOWN AS A DBA TO EVERY SITE'
FROM USER
WHERE IND_GDBA = 'YES'
  AND USER_NM IN
       (SELECT USER_NM
        FROM USER_SITE
        WHERE IND_DBA_USER = 'NO'
       )
```

DD10- Every tuple in FUNCTION is referenced by at least one FUNC_SITE-tuple for which IND_STORED = 'YES'.

```
SELECT DISTINCT 'DD10: FUNCTION IS NOT STORED AT ANY SITE'
FROM FUNCTION
WHERE (USER_NM, FUNC_NM) NOT IN
       (SELECT USER_NM, FUNC_NM
        FROM FUNC_SITE
        WHERE IND_STORED = 'YES'
       )
```

DD11- If a user-defined function operates on specific R-tables, these R-tables must be known to the site (via RT_FUNC) to which the function is assigned (via RT_SITE).

```
SELECT DISTINCT 'DD11: A FUNCTION KNOWN TO A SITE MUST BE USABLE AT THAT SITE'
FROM RT_FUNC   RTF,
     FUNCTION  FUN,
     FUNC_SITE FSI
WHERE RTF.USER_NM_FUNC = FUN.USER_NM
  AND RTF.FUNC_NM      = FUN.FUNC_NM
  AND FUN.USER_NM      = FSI.USER_NM
  AND FUN.FUNC_NM      = FSI.FUNC_NM
  AND FSI.SITE_NM NOT IN
       (SELECT SITE_NM
        FROM RT_SITE
        WHERE USER_NM = RTF.USER_NM_FUNC
          AND RT_NM   = RTF.RT_NM
       )
```

DD12- Every tuple in COND_ACT is referenced by at least one CA_SITE-tuple.

```
SELECT DISTINCT 'DD12: COND_ACT-TUPLE NOT REFERENCED BY CA_SITE-TUPLE'
FROM COND_ACT
WHERE (USER_NM, CA_NM) NOT IN
       (SELECT USER_NM, CA_NM FROM CA_SITE)
```

DD13- For every CA_SITE-tuple that references a COND_ACT-tuple that is referenced by a KEY-tuple, the columns that constitute the key (in KEY_COLUMN) must also be known to the site (in COL_SITE).

```
SELECT DISTINCT 'DD13: APPLICABILITY OF E-, R- OR ALTERNATE KEY CONSTRAINT REQUIRES
                 ALL KEY-COLUMNS TO BE KNOWN AT THE SITE'
FROM CA_SITE CAS,
     KEY
WHERE CAS.USER_NM = KEY.USER_NM
  AND CAS.CA_NM   = KEY.CA_NM
  AND EXISTS
       (SELECT * FROM KEY_COLUMN KC
        WHERE KC.USER_NM = KEY.USER_NM
          AND KC.RT_NM   = KEY.RT_NM
          AND KC.KEY_NM  = KEY.KEY_NM
          AND NOT EXISTS
                (SELECT * FROM COL_SITE CLS
                 WHERE CLS.USER_NM = KC.USER_NM
                   AND CLS.RT_NM   = KC.RT_NM
                   AND CLS.COL_NM  = KC.COL_NM
                   AND CLS.SITE_NM = CAS.SITE_NM
                )
       )
```

DD14- For every CA_SITE-tuple that references a COND_ACT-tuple that is referenced by a RT_CA-tuple, the R-tables designated by these tuples must also be known to the site (in RT_SITE).

```
SELECT DISTINCT 'DD14: APPLICABILITY OF USER-DEFINED CONDITIONAL ACTION REQUIRES ALL R-TABLES
                 TO BE KNOWN AT THE SITE'
FROM CA_SITE CAS,
     RT_CA   RTC
WHERE CAS.USER_NM = RTC.USER_NM_CA
  AND CAS.CA_NM   = RTC.CA_NM
  AND NOT EXISTS
       (SELECT * FROM RT_SITE RTS
        WHERE RTS.USER_NM = RTC.USER_NM_RT
          AND RTS.RT_NM   = RTC.RT_NM
          AND RTS.SITE_NM = CAS.SITE_NM
       )
```

DD15- For every CA_SITE-tuple that references a COND_ACT-tuple that is referenced by a COL_CA-tuple, the columns designated by these tuples must also be known to the site (in COL_SITE).

```
SELECT DISTINCT 'DD15: APPLICABILITY OF USER-DEFINED CONDITIONAL ACTION REQUIRES ALL COLUMNS TO
BE KNOWN AT THE SITE'
FROM CA_SITE CAS,
     COL_CA  CCA
WHERE CAS.USER_NM = CCA.USER_NM_CA
  AND CAS.CA_NM   = CCA.CA_NM
  AND NOT EXISTS
       (SELECT * FROM COL_SITE CLS
        WHERE CLS.USER_NM = CCA.USER_NM_RT
          AND CLS.RT_NM   = CCA.RT_NM
          AND CLS.COL_NM  = CCA.COL_NM
          AND CLS.SITE_NM = CAS.SITE_NM
       )
```

DD16- For every CA_SITE-tuple that references a COND_ACT-tuple that is referenced by a COLUMN-tuple, the columns designated by these tuples must also be known to the site (in COL_SITE).

```
SELECT DISTINCT 'DD16: APPLICABILITY OF COLUMN OR MARK CONSTRAINT REQUIRES THE COLUMN TO BE
KNOWN AT THE SITE'
FROM CA_SITE CAS,
     COLUMN  COL
WHERE CAS.USER_NM = COL.USER_NM
  AND (   CAS.CA_NM   = COL.CA_NM_C
       OR CAS.CA_NM   = COL.CA_NM_M
      )
  AND NOT EXISTS
       (SELECT * FROM COL_SITE CLS
        WHERE CLS.USER_NM = COL.USER_NM
          AND CLS.RT_NM   = COL.RT_NM
          AND CLS.COL_NM  = COL.COL_NM
          AND CLS.SITE_NM = CAS.SITE_NM
       )
```

DD17- For every COND_ACT-tuple that is referenced by a DOMAIN-tuple, the number of CA_SITE-tuples referencing it equals the number of SITE-tuples.

```
SELECT DISTINCT 'DD17: DOMAIN CONSTRAINT MUST BE ACCESSIBLE AT EVERY SITE'
FROM COND_ACT CA
WHERE (USER_NM, CA_NM) IN
       (SELECT USER_NM_CA, CA_NM
        FROM DOMAIN
        WHERE CA_NM IS NOT NULL   /* a-marked */
       )
  AND EXISTS
       (SELECT * FROM SITE
        WHERE NOT EXISTS
              (SELECT * FROM CA_SITE
               WHERE USER_NM = CA.USER_NM
                 AND CA_NM   = CA.CA_NM
                 AND SITE_NM = SITE.SITE_NM
              )
       )
```

14. DATABASE STATISTICS

ST01- The column ROWCOUNT in RT_SITE must not be I-marked if and only if IND_REPLICA = 'YES' or RL_EXPR_LG is not I-marked.

```
SELECT DISTINCT 'ST01: ROWCOUNT MUST BE I-MARKED OR MUST NOT BE I-MARKED'
FROM RT_SITE
WHERE (
          ROWCOUNT IS NULL   /* i-marked */
       AND (   IND_REPLICA = 'YES'
            OR RL_EXPR_LG IS NOT NULL  /* not i-marked */
           )
      )
   OR
      (
          ROWCOUNT IS NOT NULL     /* not i-marked */
       AND (   IND_REPLICA = 'NO'
            OR IND_REPLICA IS NULL  /* i-marked */
           )
       AND RL_EXPR_LG IS NULL       /* i-marked */
      )
```

ST02- The column DCOLCOUNT in COL_SITE must be I-marked if and only if the column ROWCOUNT in the referenced RT_SITE-tuple is also I-marked.

```
SELECT DISTINCT 'ST02: CORRESPONDING ROWCOUNTS AND DCOLCOUNTS MUST BOTH BE I-MARKED
                       OR NOT I-MARKED'
FROM RT_SITE  RTS,
     COL_SITE CLS
WHERE RTS.USER_NM = CLS.USER_NM
  AND RTS.RT_NM   = CLS.RT_NM
  AND RTS.SITE_NM = CLS.SITE_NM
  AND (
        (    RTS.ROWCOUNT  IS NULL     /*     i-marked */
         AND CLS.DCOLCOUNT IS NOT NULL /* not i-marked */
        )
       OR
        (    RTS.ROWCOUNT  IS NOT NULL /* not i-marked */
         AND CLS.DCOLCOUNT IS NULL     /*     i-marked */
        )
      )
```

ST03- The column DCOLCOUNT in COL_SITE must not exceed the column ROWCOUNT in the referenced RT_SITE-tuple.

```
SELECT DISTINCT 'ST03: DCOLCOUNT MUST NOT EXCEED CORRESPONDING ROWCOUNT'
FROM RT_SITE  RTS,
     COL_SITE CLS
WHERE RTS.USER_NM  = CLS.USER_NM
  AND RTS.RT_NM    = CLS.RT_NM
  AND RTS.SITE_NM  = CLS.SITE_NM
  AND RTS.ROWCOUNT < CLS.DCOLCOUNT
```

Appendix C A Critique of the RM/V2 Catalog Model

The aim of the manifesto is to present a first proposal for an RM/V2 catalog standard, not to assess or criticize RM/V2. Only where RM/V2 constructs or Codd's opinions on the use of RM/V2 interfere with what the authors believe to be good database design, relevant remarks have been inserted in the text.

In this appendix, an attempt is made to identify some other problem areas in RM/V2. More specifically, we are interested in issues that effect the catalog's structure. Some of these issues are well known, others are not. A number of issues have been discussed in detail in [VELD91b], albeit in a different context. The purpose of this appendix is only to identify aspects of the catalog model that may prove to be unstable, in order to focus attention on these aspects. It is not the authors' purpose to offer an in-depth analysis or to provide design alternatives at this point.

1. DOMAINS

In [CODD90], the importance of domain support is amply discussed. Unfortunately, the attention is focused at the *extended data type* aspect of the concept. The fact that a domain can also be seen as a *pool of values* is almost entirely ignored. If the catalog model provides proper support for this latter aspect, end-users can easily query the catalog to obtain information about the data values permitted in a given column. Domain and column constraints expressed by RL-statements can hardly be expected to serve the same purpose. Moreover, if the catalog can accept comment information about domains whose values are enumerated (in DOM_RANGE), it becomes possible to abolish all R-tables consisting of a primary key column and a descriptive column. Many application databases are littered with such R-tables.

As section 9 shows, semantic overrides are needed to express a number of user-defined constraints on the catalog model. It is hard to predict whether such semantic overrides are a common phenomenon in the area of application databases. If so, database designers will probably prefer RM/V2-implementations that enforce the semantic override facility less strictly.

A dubious feature of RM/V2 is the possibility to define multiple primary-non-foreign key columns on one primary domain. The authors feel that this feature, in combination with the feature that allows database designers to define multiple primary keys as the target of one foreign key, gives the database designer ample opportunity for suboptimal database designs. The examples in [CODD90] in which the feature is exploited are clear examples of such designs (see [VELD91b]).

2. KEYS

As discussed above, the authors feel that a foreign key should not be allowed to reference more than one primary key. Codd's definition of referential integrity is also too lenient in the fact that it allows columns that constitute a composite foreign key to be partly A-marked. If, as the authors believe, this facility is rarely needed, the database designer has to specify an additional user-defined constraint asserting that either all key columns that constitute a foreign key are A-marked or

unmarked.

RM/V2 forbids foreign key columns to be I-marked (constraint BM14) because "*... such a mark contradicts the foreign key concept*" [p. 175]. The authors feel that Codd's position is debatable because non-foreign key columns can easily become foreign key columns. Take for example the R-tables EMP and JOB that contain information about a companies employees and their jobs:

EMP(EMP#, ..., JOB#, CAR#)
JOB(JOB#, ..., COMP_CAR_INDICATOR)

It is assumed that a company car is assigned to employees performing certain jobs. It seems thus appropriate to assign an I-mark to the column CAR# if the EMP-tuple references a JOB-tuple for which COMP_CAR_INDICATOR equals 'NO', while an A-mark will be assigned to CAR# if COMP_CAR_INDICATOR equals 'YES' and the employee's car registration number is currently unknown.

Now suppose that the database is to contain car data other than the car's registration number. If the database is to remain in 5NF, an extra R-table will be added to the database with CAR# as its primary key. It is very hard to see why the column CAR# in EMP, that now constitutes a foreign key, cannot accept I-marks any more.

3. CONDITIONAL ACTIONS

In section 7, some problems with respect to the treatment of conditional actions have been discussed. Constraint classification is in the authors' view a very important topic. Codd's reliance on RL as the only tool for expressing constraints leads to a number of problems.

First, catalog supported constraint classification prevents the database designer from overlooking relevant constraints. One rarely encounters database designs in which referential integrity constraints have been overlooked, while overlooking user-defined constraints is a common phenomenon.

Second, catalog supported constraint classification prevents the designer from coding a large number of constraints. For instance, if all constraints of types E, R, and D[1] were expressed in SQL in appendix B, the appendix would be much longer. Instead, the form of these constraints can be inferred from the catalog's contents by programmers and application users as well as by the RDBMS itself. The RDBMS can easily translate these constraints to RL-statements in order to enforce them.

Finally, catalog supported constraint classification makes it easier for both the RDBMS and the database designer to interpret the semantics represented by the database structure. Few designers will question the statement that a version of relational model without entity and referential integrity would lower their productivity and the quality of their work. If so, classifying more constraints will improve productivity and quality of database design. Obviously (and unfortunately), the law of diminishing returns and the subjective nature of what constraints are liable for classification pose large problems in this respect. Perhaps it is best to leave it to RDBMS- and dictionary-vendors to provide support for generalized constraints that, in their opinion, occur frequently. Support

---

[1] The D-type constraints referred to are those constraints that are fully specified by the contents of their DOM_RANGE tuples.

for such constraints can then be supported by the catalog model with some specific extensions.

A very difficult problem is caused by the possibility that a conditional action triggers other conditional actions. If a conditional action triggers database updates, it is quite possible to inadvertently generate deadlock situations. Moreover, the sequence in which the constraints are executed may influence the database state after the transaction. Codd himself is not very optimistic about automated support for detection of these situations [p. 266]. The authors' opinion is that the prospect of a future in which Database *Management* Systems become unmanageable trigger mechanisms is not very appealing.

The MESDAG Research Group

INTRODUCTION

The MESDAG project is a joint project endorsed by three organizations in the Netherlands: the N.V. Nederlandse Spoorwegen (The Netherlands Railways Company), RAET N.V. and the Vrije Universiteit of Amsterdam. The MESDAG project originated at RAET N.V. during the second half of 1989 as an outgrowth of research done in the field of active data dictionary models. This research and a prototype of an active data dictionary form the basis for the mission of the MESDAG project that officially started its activities in September 1990.

MESDAG is an abbreviation of:

MEta Systems Design And Generation

MISSION AND OBJECTIVES

The mission of the MESDAG project is to prove the feasibility of developing inherently flexible information systems by introducing higher levels of logical data independence.

Derived from this mission following are the two main objectives:

1. Examine the feasibility and initiate the development of an active, self-referential data dictionary model in which both a description of the database data and a description of all specifiable application design data can be stored. This data dictionary model should contain sufficient semantic aspects (like domains, constraints and time aspects) to assure the integrity, consistency and validity of the stored (meta) data, to avoid maintenance and to support query-formulation independent of current database structure.

2. Examine the feasibility and initiate the development of the possibilities of data dictionaries in general and the described data dictionary in specific. This analysis of possibilities is directed at the embedding in and developing methods, techniques, methodologic guidelines and automated tools for the design, implementation and maintenance of flexible information systems.

MEMBERS OF THE MESDAG RESEARCH GROUP

1. Dr. E.R.K. Spoor
Dr. E.R.K. Spoor is associate professor at the Vrije Universiteit Amsterdam. He teaches and consults in the area of database systems and database development with a focus on the use of these technologies in organizations. His eighteen years of experience with computer technology includes eight years with NCR and six years with the Vrije Universiteit, first as a systems engineer and later as a computer scientist. He is one of the founders and board members of two automation oriented organizations: PSB (Amsterdam) and VDA (Hilversum).

2. Drs. R.J. Veldwijk
Drs. R.J. Veldwijk graduated from the Vrije Universiteit Amsterdam in 1986. In his quality as consultant at RAET N.V. Utrecht, he is among others responsible for the design and implementation of data models. His main interest lies in developing and implementing self-knowledgeable database models, aimed at reducing maintenance costs and at improving the accessibility of databases by end-users. Furthermore he teaches courses in data modelling.

3. Drs. M. Boogaard
Drs. M. Boogaard is assistant researcher at the Vrije Universiteit Amsterdam. Furthermore, he is part-time involved in projects by the Netherlands Railways Company. He graduated from the Vrije Universiteit Amsterdam, in August 1990. The objective of his research is to develop an approach to achieve higher levels of logical data independence for both end-users and application programs and to analyze the consequences of the level of logical data independence accomplished on the system development life cycle in general and on software maintenance and database inquiry in particular.

Guest co-author dr. R.B. Buitendijk
After completing the study of Computer Science at the Vrije Universiteit Amsterdam in 1987, dr. R.B. Buitendijk received his PhD in April 1991. He is currently employed by the Netherlands Railways Company as a consultant primarily focused on databases, data management, data modelling, and CASE-tools.

ACCOMMODATION ADDRESS

Vrije Universiteit
Faculteit Economie & Econometrie
Vakgroep BIK
De Boelelaan 1105
1081 HV Amsterdam Phone: +31-20-548708
The Netherlands Fax : +31-20-6462645

1991-1	N.M. van Dijk	On the Effect of Small Loss Probabilities in Input/Output Transmission Delay Systems
1991-2	N.M. van Dijk	Letters to the Editor: On a Simple Proof of Uniformization for Continious and Discrete-State Continious-Time Markov Chains
1991-3	N.M. van Dijk P.G. Taylor	An Error Bound for Approximating Discrete Time Servicing by a Processor Sharing Modification
1991-4	W. Henderson C.E.M. Pearce P.G. Taylor N.M. van Dijk	Insensitivity in Discrete Time Generalized Semi-Markov Processes
1991-5	N.M. van Dijk	On Error Bound Analysis for Transient Continuous-Time Markov Reward Structures
1991-6	N.M. van Dijk	On Uniformization for Nonhomogeneous Markov Chains
1991-7	N.M. van Dijk	Product Forms for Metropolitan Area Networks
1991-8	N.M. van Dijk	A Product Form Extension for Discrete-Time Communication Protocols
1991-9	N.M. van Dijk	A Note on Monotonicity in Multicasting
1991-10	N.M. van Dijk	An Exact Solution for a Finite Slotted Server Model
1991-11	N.M. van Dijk	On Product Form Approximations for Communication Networks with Losses: Error Bounds
1991-12	N.M. van Dijk	Simple Performability Bounds for Communication Networks
1991-13	N.M. van Dijk	Product Forms for Queueing Networks with Limited Clusters
1991-14	F.A.G. den Butter	Technische Ontwikkeling, Groei en Arbeidsproduktiviteit
1991-15	J.C.J.M. van den Bergh, P. Nijkamp	Operationalizing Sustainable Development: Dynamic Economic-Ecological Models
1991-16	J.C.J.M. van den Bergh	Sustainable Economic Development: An Overview
1991-17	J. Barendregt	Het mededingingsbeleid in Nederland: Konjunktuurgevoeligheid en effektiviteit
1991-18	B. Hanzon	On the Closure of Several Sets of ARMA and Linear State Space Models with a given Structure
1991-19	S. Eijffinger A. van Rixtel	The Japanese Financial System and Monetary Policy: a Descriptive Review
1991-20	L.J.G. van Wissen F. Bonnerman	A Dynamic Model of Simultaneous Migration and Labour Market Behaviour